PENGEMBANGAN APLIKASI PUZZLE GAME HALMA MENGGUNAKAN J2SE

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Teknik Informatika





Oleh :

Nama : Sasongko Ari Wibowo

No. Mahasiswa : 01 523 182

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA

2007

LEMBAR PENGESAHAN PEMBIMBING

PENGEMBANGAN APLIKASI PUZZLE GAME HALMA MENGGUNAKAN J2SE

TUGAS AKHIR



oleh:

Nama : Sasongko Ari Wibowo

No. Mahasiswa : 01 523 182

Yogyakarta, 28 Agustus 2007

Pembimbing

Taufiq Hidayat, ST.,MCS.

LEMBAR PENGESAHAN PENGUJI

PENGEMBANGAN APLIKASI PUZZLE GAME HALMA MENGGUNAKAN J2SE

TUGAS AKHIR

oleh:

Nama : Sasongko Ari Wibowo

No. Mahasiswa : 01 523 182

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 28 Agustus 2007

Tim Penguji

Taufik Hidayat, ST., MCS. _____

Hendrik, ST. _____

Nur Wijayaning Rahayu, S.Kom. _____

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia

★ YOGYAKARTA ★

Yudi Prayudi, S.Si., M.Kom.

HALAMAN PERSEMBAHAN



Kupersembahkan setitik pencapaian awal dari masa depanku dengan penuh cinta dan kerendahan hati

- Sembahanku, **Allah SWT.**

  Ketulusan cinta-Mu padaku membuatku mampu berdiri tegak di tengah-tengah badai kehidupan yang aku lalui.

  Tiada kata yang mampu menggantikan "berlian-berlian" yang selalu Engkau berikan di setiap hembusan nafasku.

- Suri Tauladanku, **Rasulullah Muhammad Saw.**

  Keluhuruan budi pekerti dan ketegaranmu memberi contoh yang baik bagiku beserta seluruh hamba Allah benar-benar menggugah semangatku untuk pantang menyerah menghadapi seluruh permasalahan hidup ini

- **Bapak Ibuku** tercinta yang penuh kesabaran, keikhlasan, ketulusan dalam membimbingku hingga aku bisa seperti ini.

  Pengorbanan kalian tak akan pernah kusia-siakan karena dari pengorbanan kalianlah aku termotivasi untuk membuat kalian tersenyum dan bangga dengan apa yang kuraih.

- ***Sahabat-sahabat BINER Informatika 01**, Agus-Munir, Iis, ,Towi, Lucky, Ludfi, Adit, Mada, Bol, Irul, Hasan, Nhal, Mamad, Black, Rindang, Sigit, Pahe, Wulan, Budhe, Ipul, Ayu, dan temen-temen yang lainnya makasih banget atas semangatnya dan hangatnya kebersamaan kita. Thanx 4 U all guyz . . .*
- ***Bagian Pengajaran FTI UII, Perpustakaan fakultas FTI UII, Jurusan FTI UII** ... terimakasih atas kerjasamanya.*
- ***Semua komunitas INFORMATIKA FTI UII semua angkatan dan FTI UII** ... tempat yang nyaman dan membanggakan ...*
- *Dan semua orang yang telah banyak berperan dalam hidupku.*
- *Dan semua Sahabat-sahabatku yang lainnya, semoga persahabatan kita tidak akan pernah berakhir walau badai menerjang. Mengenal kalian membuatku mengerti arti sebuah persahabatan sejati. Sahabat sejati selalu bersama dalam suka dan duka.*

HALAMAN MOTTO

Allah tidak membebani seseorang, melainkan sesuai dengan kesanggupannya "

(QS. Al Baqarah : 256)

" Sesungguhnya di samping kesukaran terdapat kemudahan "

(QS. Al-Insyirah : 5)

Hidup ini cuma sementara...
Janganlah kau sia-sia kan waktumu untuk hal-hal yang tidak berguna

Hidup cuma sekali...
Hiasilah hidupmu dengan senyuman

Tetaplah tersenyum & bersyukur_
Karena apa yang kau dapat adalah pemberian terbaik Allah untukmu

Optimislah dengan apa yang akan kau lakukan karena pesimis hanya akan membuat kegagalan

Jika engkau kesulitan dalam pekerjaanmu, janganlah berputus asa, janganlah gelisah, dan janganlah ragu. Percayalah, jalan keluar akan segera datang

Janganlah merasa menjadi orang terbaik, karena pasti ada orang yang lebih baik dari kita, dan di atas langit masih ada langit

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

اَلسَّلَامُ عَلَيْكُمْ وَرَحْمَةُ اللَّهِ وَبَرَكَاتُهُ

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT. yang senantiasa memberikan rahmat dan hidayah-Nya kepada kita semua, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "PENGEMBANGAN APLIKASI PUZZLE GAME HALMA MENGUNAKAN J2SE". Sholawat serta salam tak lupa tercurah kepada junjungan kita Rasulullah Muhammad SAW., keluarga, sahabat, serta pengikutnya hingga akhir zaman.

Tugas Akhir ini diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana Teknik Informatika.

Kemudian atas terselesaikannya Tugas Akhir ini, penulis mengucapkan banyak terima kasih kepada semua pihak yang telah bersedia memberikan bantuan serta motivasinya selama ini. Untuk itu penulis tidak lupa menyampaikan rasa terimakasih kepada :

1. Allah SWT, Tuhan Semesta Alam.
2. Ayah Ibu tercinta yang telah sudi merawat amanat dari Sang Khalik serta keluarga, terima kasih atas kasih sayang yang tulus, doa dan semangat baik moral maupun material yang diberikan, sungguh kami tidak akan bisa membalas semua kebaikan, ketulusan serta kasih sayang yang kalian berikan, biarlah Allah SWT. yang akan membalasnya.

3. Taufiq Hidayat, ST., MCS. Selaku Dosen pembimbing yang telah mengarahkan dan membimbing dalam pembuatan laporan dari awal hingga selesainya laporan Tugas Akhir ini.
4. Bapak Yudi Prayudi, SSi, Mkom. selaku Ketua Jurusan Teknik Informatika, Universitas Islam Indonesia.
5. Bapak dan Ibu dosen Teknik Informatika UII yang telah memberikan ilmu serta mendidik penulis selama menempuh studi di Fakultas Teknologi Industri, Universitas Islam Indonesia.
6. Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu sehingga laporan ini dapat terselesaikan.

Semoga Allah SWT melimpahkan pahala-Nya kepada semua pihak yang telah membantu terselesaikannya penulisan laporan Tugas Akhir ini sesuai dengan amal kebaikannya.

Penulis menyadari bahwa dalam penyusunan laporan Tugas Akhir ini masih banyak terdapat kekurangan yang jauh dari sempurna. Untuk itu penulis tidak menutup kemungkinan kritik dan saran yang bersifat membangun guna penyempurnaan hasil Tugas Akhir ini di masa mendatang.

Akhir kata semoga laporan ini dapat bermanfaat bagi penyusun dan semua yang berkenan membacanya.

وَالسَّلَامُ عَلَيْكُمْ وَرَحْمَةُ اللهِ وَبَرَكَاتُهُ

Yogyakarta, 28 Agustus 2007

Penyusun

ABSTRAKSI

Dalam bidang komputer grafik, banyak bidang yang bisa dikembangkan, salah satunya adalah pembuatan aplikasi *game*. *Game* mempunyai banyak jenis atau *genre,* salah satunya adalah *puzzle games. Game* berjenis *puzzle* adalah *game* yang kebanyakan berbentuk mini *game* atau *game* ringan, tidak memiliki alur cerita yang berat, dan diperlukan sedikit berpikir dalam menyelesaikan permasalahan atau tantangan di dalam *game* tersebut.

Halma adalah salah satu *game* berjenis *puzzle.* Halma (yang dalam bahasa Yunani berarti lompat) adalah permainan yang ditemukan tahun 1883 oleh seorang dokter bedah asal Inggris, George Howard Monks. Permainan ini dimainkan oleh tiga orang pemain di atas sebuah papan permainan berbentuk bintang. Tujuan memenangkan permainan halma adalah menyeberangkan bidak-bidak halma milik pemain ke tempat yang ada di seberang. Setiap pemain berjalan bergiliran dimana pemain tersebut hanya bisa menggerakkan satu bidak halmanya pergiliran. Pemenang permainan halma adalah pemain yang paling cepat memindahkan semua bidak halmanya ke tempat tujuan di seberang.

Pembuatan aplikasi *game* halma ini menggunakan teknologi J2SE. Pada aplikasi ini permainan dapat dimainkan oleh tiga pemain, yaitu kombinasi antara pemain manusia dan komputer atau seluruhnya pemain manusia atau komputer. Bidak halma yang tidak dimainkan oleh pemain manusia akan dimainkan oleh komputer dimana komputer akan mencari rute optimum bidaknya menggunakan kecerdasan buatan metode *Generate and Test*. Jika salah satu pemain telah menyelesaikan permainannya maka pemain tersebut menjadi pemenang pertama dan dua pemain yang tersisa melanjutkan permainan sampai didapat pemenang kedua.

Kata kunci: Java, J2SE, *Generate and Test,* Halma.

TAKARIR

Generate and Test

Suatu metode pencarian heuristic yang paling sederhana. Merupakan gabungan dari metode *depth-first search* dan *backtracking.*

Unified Modelling Language (UML)

Bahasa grafis untuk mendokumentasikan, menspesifisikasikan, dan membangun sistem perangkat lunak.

Game Puzzle

Jenis permainan membutuhkan pemain untuk berpikir dan menyelesaikan suatu masalah yang diberikan atau untuk menuju tujuan.

Game worlds

Elemen-elemen utama yang terdapat dalam suatu program game.

Garbage collector

Fasilitas dari program Java untuk mengurangi beban pengelolaan memori oleh programmer dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat di dalam bahasa yang memungkinkan alokasi dinamis.

Graphical User Interface (GUI)

Antarmuka komputer yang berbasiskan grafis. Merupakan citra grafis yang ditampilkan di layar komputer untuk mengakses aplikasi software.

Application Programming Interface (API)

Penghubung suatu aplikasi dengan aplikasi lainnya. Suatu rutin standar yang memungkinkan programmer menggunakan system function.

Game board

Layanan aplikasi game berupa bentuk tampilan game beserta latar belakangnya.

Java Development Kit (JDK)

Sekumpulan program dan library Java yang digunakan untuk menjalankan dan mengembangkan program Java.

DAFTAR ISI

HALAMAN JUDUL..... i
LEMBAR PENGESAHAN PEMBIMBING..... ii
LEMBAR PENGESAHAN PENGUJI..... iii
HALAMAN PERSEMBAHAN..... iv
HALAMAN MOTTO..... vii
KATA PENGANTAR..... viii
ABSTRAKSI..... x
TAKARIR..... xi
DAFTAR ISI..... xii
DAFTAR TABEL..... xvi
DAFTAR GAMBAR..... xvii

BAB I **PENDAHULUAN**..... 1
1.1 Latar Belakang..... 1
1.2 Rumusan Masalah..... 2
1.3 Batasan Masalah..... 2
1.4 Tujuan Penelitian..... 3
1.5 Manfaat Penelitian..... 3
1.6 Metodologi Penelitian..... 3
1.7 Sistematika Penulisan..... 4

BAB II **LANDASAN TEORI**..... 5
2.1 Pendekatan Berorientasi Objek..... 5
2.1.1 Objek..... 5

2.1.2 Kelas..... 6

2.1.2.1 Kegunaan Kelas..... 6

2.1.2.2 Karakteristik Kelas..... 7

2.1.2.3 Properti Kelas..... 8

2.2 Teknologi Java..... 9

2.2.1 Java 2..... 9

2.3 Teori Dalam Pembuatan Program Game..... 11

2.4 Permainan Halma..... 13

2.5 Pencarian Metode *Generate and Test*..... 14

BAB III METODOLOGI PENELITIAN..... 16

3.1 Analisis Kebutuhan..... 16

3.1.1 Metode Analisis..... 16

3.1.2 Hasil Analisis..... 16

3.2 Perancangan Perangkat Lunak..... 19

3.2.1 Metode Perancangan..... 19

3.2.2 Desain Sistem..... 20

3.2.2.1 Diagram *Use-Case*..... 21

3.2.2.2 Diagram Kelas..... 22

3.2.2.3 Diagram Sekuen..... 26

3.2.2.4 Diagram Aktivitas..... 34

3.2.3 Rancangan Antarmuka..... 35

3.2.3.1 Rancangan Antarmuka Jendela Utama..... 35

3.2.3.2 Rancangan Antarmuka Jendela Pengaturan 36

3.2.3.3 Rancangan Antarmuka Jendela Cara Bermain..... 37
3.2.3.4 Rancangan Antarmuka Jendela About..... 38
3.3 Implementasi Perangkat Lunak..... 38
3.3.1 Implementasi..... 39
3.3.1.1 Implementasi Sistem..... 39
3.3.1.1.1 Implementasi Konsep *Game* Halma Pada Sistem..... 39
3.3.1.1.2 Penyimpanan Data Koordinat dan Konversi Koordinat – Indeks..... 46
3.3.1.1.3 Pengesetan Awal Game Baru..... 48
3.3.1.1.4 Pengesetan Giliran..... 49
3.3.1.1.5 Prosedur Pemain Komputer Bermain..... 51
3.3.1.1.6 Prosedur Pemain Manusia Bermain..... 56
3.3.1.1.7 *Method cekValid*..... 58
3.3.1.1.8 Pemindahan Bidak..... 60
3.3.1.1.9 Pengecekan dan Pengesetan Pemenang... 61
3.3.1.2 Implementasi Antarmuka..... 63
3.3.1.2.1 Antarmuka Jendela Utama..... 63
3.3.1.2.2 Antarmuka Jendela Pengaturan..... 64
3.3.1.2.3 Antarmuka Jendela Cara Bermain..... 65
3.3.1.2.4 Antarmuka Jendela About..... 66

BAB IV ANALISIS KINERJA PERANGKAT LUNAK..... 67

4.1 Pengujian dan Analisis..... 67

4.2 Hasil Analisis..... 67

4.2.1 Analisis Kesesuaian..... 68

4.2.2 Analisis Tingkat Kesulitan Komputer..... 69

4.2.3 Analisis Perbandingan..... 71

4.2.4 Kelebihan dan Kekurangan Program..... 71

BAB V PENUTUP..... 73

5.1 Kesimpulan..... 73

5.2 Saran..... 74

DAFTAR PUSTAKA..... 75

DAFTAR TABEL

Tabel 4.1 Tabel hasil analisis 68

Tabel 4.2 Tabel hasil perbandingan..... 71

DAFTAR GAMBAR

Gambar 2.1	Pembagian lingkungan kerja pada teknologi Java.....	11
Gambar 2.2	Papan permainan halma.....	14
Gambar 3.1	Diagram *use-case*.....	21
Gambar 3.2	Diagram Kelas	26
Gambar 3.3	Diagram sekuen pemanggilan game.....	28
Gambar 3.4	Diagram sekuen pengaturan.....	29
Gambar 3.5	Diagram sekuen *game* baru.....	30
Gambar 3.6	Diagram sekuen manusia menjalankan bidak.....	31
Gambar 3.7	Diagram sekuen komputer menjalankan bidak...	32
Gambar 3.8	Diagram sekuen cara bermain.....	33
Gambar 3.9	Diagram sekuen *about*.....	33
Gambar 3.10	Diagram aktivitas.....	34
Gambar 3.11	Rancangan antarmuka jendela utama.	36
Gambar 3.12	Rancangan antarmuka jendela pengaturan.....	37
Gambar 3.13	Rancangan antarmuka jendela cara bermain.....	37
Gambar 3.14	Rancangan antarmuka jendela *about*.....	38
Gambar 3.15	Arah perjalanan pemain.....	40
Gambar 3.16	Koordinat posisi bidak biru.....	41
Gambar 3.17	Koordinat posisi bidak merah.....	42
Gambar 3.18	Koordinat posisi bidak hijau.....	42
Gambar 3.19	Tampilan antarmuka jendela utama.....	64

Gambar 3.20 Tampilan antarmuka jendela pengaturan..... 65

Gambar 3.21 Tampilan antarmuka jendela cara bermain..... 66

Gambar 3.22 Tampilan antarmuka jendela *about*..... 66

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam bidang komputer grafik, banyak bidang yang bisa dikembangkan, salah satunya adalah pembuatan aplikasi *game*. *Game* mempunyai banyak jenis atau *genre,* salah satunya adalah *puzzle games* [GOD98]. *Game* berjenis *puzzle* adalah *game* yang kebanyakan berbentuk mini *game* atau *game* ringan, tidak memiliki alur cerita yang berat, dan diperlukan sedikit berpikir dalam menyelesaikan permasalahan atau tantangan di dalam *game* tersebut. Karena sifatnya yang ringan maka *game* jenis ini memiliki ruang lingkup pengguna yang luas, mulai dari anak-anak sampai orang dewasa, baik laki-laki maupun perempuan.

Halma adalah salah satu *game* berjenis *puzzle.* Halma (yang dalam bahasa Yunani berarti lompat) adalah permainan yang ditemukan tahun 1883 oleh seorang dokter bedah asal Inggris, George Howard Monks. Permainan ini dimainkan oleh tiga orang pemain di atas sebuah papan permainan berbentuk bintang. Tujuan memenangkan permainan halma adalah menyeberangkan bidak-bidak halma milik pemain ke tempat yang ada di seberang. Setiap pemain berjalan bergiliran dimana pemain tersebut hanya bisa menggerakkan satu bidak halmanya pergiliran. Pemenang permainan halma adalah pemain yang paling cepat memindahkan semua bidak halmanya ke tempat tujuan di seberang. Agar bisa berjalan dengan cepat, pemain boleh memindahkan bidak halma dengan cara

melompati bidak yang lain, baik itu milik pemain sendiri maupun milik lawan dengan arah yang lurus. Begitu bidak milik pemain melompati bidak lainnya, bila ada kesempatan melompat yang lainnya, pemain bisa melompat lagi sampai tidak ada jalan lagi [SYA05].

Pada aplikasi *game* halma ini permainan dapat dimainkan oleh tiga pemain, yaitu kombinasi antara pemain manusia dan komputer atau seluruhnya pemain manusia atau komputer. Bidak halma yang tidak dimainkan oleh manusia akan dimainkan oleh komputer menggunakan kecerdasan buatan. Jika salah satu pemain telah menyelesaikan permainannya maka pemain tersebut menjadi pemenang pertama dan dua pemain yang tersisa melanjutkan permainan sampai didapat pemenang kedua.

1.2 Rumusan Masalah

Masalah yang ditimbulkan dari pembuatan *software* ini adalah :

"Bagaimana mengembangkan aplikasi *game* puzzle halma yang dilengkapi kecerdasan buatan menggunakan J2SE?".

1.3 Batasan Masalah

Batasan-batasan masalah dalam pembuatan *software* ini adalah :

1. *Software* ini dikembangkan menggunakan teknologi J2SE.
2. *Software* ini berjalan pada komputer *stand alone*, dan tidak dapat digunakan pada jaringan.

3. Komputer menggunakan kecerdasan buatan untuk menentukan langkah menuju tujuan menggunakan pencarian rute optimum, tetapi tidak bisa memperhitungkan apakah langkah yang dilakukannya justru menguntungkan bagi lawan.

1.4 Tujuan Penelitian

Tujuan penelitian adalah :

Mengembangkan sebuah aplikasi puzzle *game* Halma yang dapat dimainkan manusia maupun komputer yang dilengkapi dengan kecerdasan buatan.

1.5 Manfaat Penelitian

Manfaat penelitian adalah :

Memberikan alternatif hiburan dalam bentuk *game* halma kepada para pengguna komputer

1.6 Metodologi Penelitian

Untuk melancarkan penelitian tersebut, maka perlu dilakukan hal-hal sebagai berikut :

1. Analisis kebutuhan, meliputi penganalisisan hal-hal yang dibutuhkan dalam pembuatan program ini.
2. Perancangan, meliputi pembuatan desain yang akan diimplementasikan pada program ini.
3. Implementasi, meliputi realisasi rancangan dalam bentuk program.

1.7 Sistematika Penulisan

Sistem penulisan laporan berguna untuk memberi gambaran umum dari keseluruhan isi laporan. Penulisan laporan ini dibagi dalam beberapa bab, yaitu :

Bab I berisi pendahuluan yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

Bab II berisi landasan teori yang meliputi gambaran umum tentang teori yang diterapkan dalam pembuatan *software* ini. Dalam bab ini dijelaskan tentang teori-teori pemrograman berorientasi objek, Teknologi java, pemrograman java pada perangkat lunak, teori pembuatan program *game*, penjelasan mengenai *game* halma dan teori algoritma *generate and test.*

Bab III berisi metodologi penelitian meliputi analisis kebutuhan yang berisi hasil analisis hal-hal yang dibutuhkan dalam pembuatan program, perancangan yang berisi pembuatan desain yang akan diimplementasikan pada program, dan implementasi yang berisi realisasi rancangan dalam bentuk program.

Bab IV berisi analisis kinerja perangkat lunak.

Bab V berisi penutup yang meliputi kesimpulan dan saran.

BAB II

LANDASAN TEORI

2.1 Pendekatan Berorientasi Objek

Pendekatan berorientasi objek adalah cara memandang persoalan menggunakan model-model yang diorganisasikan seputar konsep objek yang mengkombinasikan struktur data dan perilaku entitas.

Pada pendekatan berorientasi objek, blok pembangun utama sistem perangkat lunak adalah objek dan/atau kelas. Kandidat-kandidat kelas umumnya merupakan sesuatu yang diperoleh dari kosakata di ruang persoalan dan/atau ruang solusi. Kelas merupakan deskripsi himpunan objek yang serupa. Setiap objek memiliki identitas (sesuatu yang membedakan dengan objek lain), *state* (data yang diasosiasikan dengannya), dan perilaku [RIC03].

Adanya pewarisan dalam pendekatan berorientasi objek memungkinkan struktur dan perilaku dipakai bersama dalam beberapa subklas serupa tanpa mengalami redundansi dan penggunaan ulang (*reuse*) rancangan dan kode pada proyek-proyek berikutnya.

2.1.1 Objek

Objek adalah benda, baik yang berwujud nyata, maupun yang tidak nyata, seperti halnya yang berupa sistem atau konsep. Dalam pendekatan berorientasi objek, objek adalah unit terkecil pemrograman yang masih memiliki data (sifat karakteristik) dan fungsi. [RIC03]

2.1.2 Kelas

Kelas merupakan deskripsi satu objek atau lebih dengan sekumpulan atribut dan layanan seragam, termasuk deskripsi penciptaan objek baru di kelas itu. Kelas objek mendeskripsikan kelompok objek dengan properti-properti serupa, perilaku yang umum, kelompok struktur data yang dibentuk oleh kelas, keterhubungan terhadap himpunan objek-objek lain dan semantik yang umum. [RIC03]

2.1.2.1 Kegunaan Kelas

Kegunaan kelas adalah sebagai berikut :

a. Kelas merupakan abstraksi dari masalah yang dihadapi. Abstraksi memberikan kemampuan generalisasi beberapa kasus spesifik menjadi induk kasus-kasus yang serupa.
b. Definisi nama kelas dan atribut, dan operasi-operasi cukup disimpan dan ditulis sekali per kelas. Semua objek dalam kelas mendapatkan manfaat penggunaan kembali (*reuse*) kode itu.
c. Kelas merupakan penerapan prinsip pengkapsulan yang memadukan properti-properti unit komputasi (data dan operasi) untuk mengembangkan satu tanggungjawab tertentu yang diembannya.

2.1.2.2 Karakteristik Kelas

Kelas memiliki karkteristik-karakteristik yang unik yaitu :

a. *Tangibility*

Kemampuan kelas untuk merepresentasikan sesuatu yang terdapat entitas fisiknya atau informasi abstrak.

b. *Inclusiveness*

Kelas dapat berisi kelas-kelas lain dalam satu kelas (kelas agregat) atau kelas tidak berisi kelas-kelas lain (kelas atomik).

c. *Sequentially*

Kelas dapat memiliki banyak alur kendali (kelas konkuren) atau kelas dikendalikan dari sumberdaya eksternal (kelas sekuen).

d. *Persistence*

Sifat kelas berdasar *persistence* terbagi sebagai berikut :

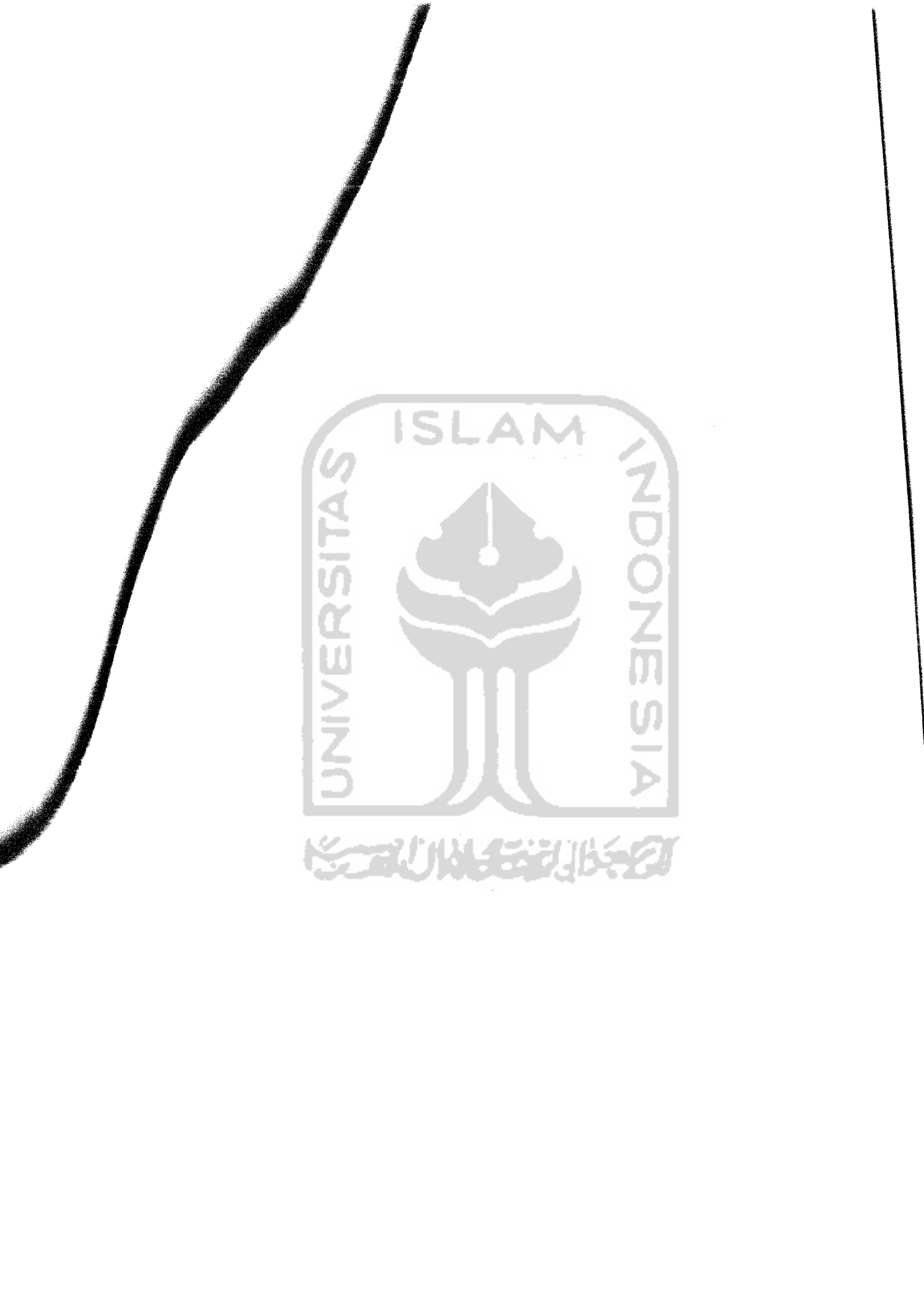
(i). *Transient* , yaitu diciptakan dan dihilangkan semua operasi program.

(ii). *Temporary,* yaitu diciptakan selam operasi dan dihilangkan ketika program berakhir.

(iii). *Permanent,* yaitu disimpan di basisdata.

e. *Integrity*

Kelas dapat dikorupsi, yaitu sumber dayanya tidak dijaga dari pengaruh luar, atau kelas *guarded,* yaitu kelas memaksakan pengendalian pengaksesan sumberdaya di objek.



2.1.2.3 Properti Kelas

Kelas memiliki properti-properti berikut :

a. Nama

Nama digunakan untuk membedakan dengan kelas-kelas lain.

b. Atribut

Atribut adalah properti bernama di kelas yang mendeskripsikan satu *range* nilai yang dapat dimiliki instan kelas dan merepresentasikan suatu properti dari sesuatu yang dimodelkan. Kelas dapat memiliki beberapa atribut atau tidak sama sekali.

c. Operasi/Metode

Operasi adalah implementasi layanan yang dapat diminta dari suatu objek. Kelas dapat memiliki beberapa objek atau tidak sama sekali.

d. Tanggungjawab (*responsibilities*)

Tanggungjawab adalah kontrak atau kewajiban kelas. Kelas yang berisi sejumlah atribut dan operasi merupakan manifestasi tanggungjawab yang diembannya untuk dilaksanakan.

2.2 Teknologi Java

Java adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh *Sun Microsystems.* Sebagai bahasa pemrograman, Java dikenal sebagai bahasa pemrograman tingkat tinggi dengan fitur-fitur utama sebagai berikut [HAR04] :

a. *Simple*

Java dirancang untuk mudah dipelajari, terutama bagi yang telah mengenal C/C++ akan mudah untuk berpindah ke Java.

b. *Object Oriented*

Java adalah bahasa pemrograman berorientasi objek yang merupakan pemrograman masa depan.

c. *Robust and Secure*

Java dirancang sebagai bahasa pemrograman yang handal dan aman. Manajemen memori dan pengaturan keamanan yang bagus merupakan keunggulan bahasa ini.

d. *Architecture Neutral and Portable*

Java dirancang untuk dapat dijalankan di semua platform, tanpa peduli apakah arsitektur perangkat Intel, AMD atau Machintos. Java memungkinkan untuk berjalan pada sistem operasi apapun, baik di Windows, Linux, Solaris, MacOS dan sebagainya tanpa harus melakukan kompilasi ulang pada program Java yang dibuat.

e. *High Perfomance*

Java dirancang untuk menghasilkan aplikasi-aplikasi yang terbaik.

2.2.1 Java 2

Setelah mengalami pengembangan-pengembangan, Sun Microsystem akhirnya mengenalkan Java versi terbaru yaitu Java versi 1.2 atau lebih dikenal dengan nama Java 2 yang terdiri atas JDK dan JRE versi 1.2. Aplikasi-aplikasi

yang mendukung dan kompatibel dengan Java 2 dikenal sebagai Java 2 Compliant. Terdapat tiga kategori dalam Java 2 ini, yaitu :

1. Java 2 Standard Edition (J2SE)

   Kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada level PC (Personal Computer).

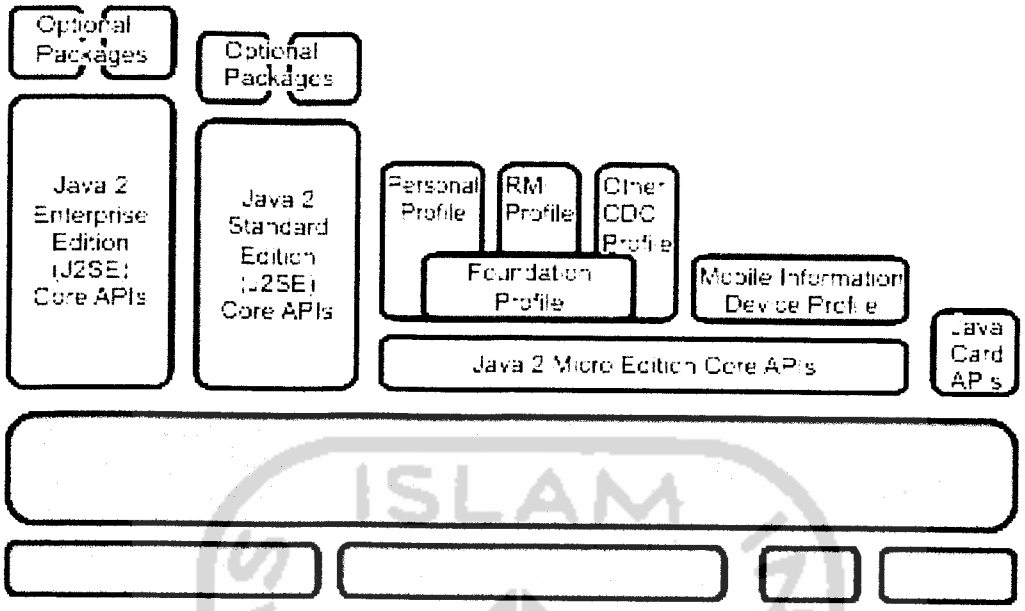
2. Java 2 Enterprise Edition (J2EE)

   Kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada lingkungan enterprise dengan menambah fungsionalitas-fungsionalitas Java semacam EJB (Enterprise Java Bean), Java CORBA, Servlet dan JSP, serta Java XML (Extensible Markup Language).

3. Java 2 Micro Edition (J2ME)

   Kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada *handheld devices* atau perangkat-perangkat mobile lainnya semacam *handphone*, Palm, PDA dan Pocket PC.

Pembagian lingkungan kerja pada teknologi Java terlihat pada Gambar 2.1.



Gambar 2.1 Pembagian lingkungan kerja pada teknologi Java

2.3 Teori Dalam Pembuatan Program Game

Urutan pembuatan *game* dan pengembangan program *game* adalah sebagai berikut [HEN89] :

a. Tentukan tipe game yang ingin dibuat

Penentuan ini sebagai dasar mulai bekerja sampai mendapatkan ide yang bagus untuk dibuat program game.

b. Definisikan model game dan tujuannya

Pada tahap ini sebaiknya model game ditulis secara jelas sehingga jika terjadi perubahan maka gamenya tetap konsisten dan tidak membingungkan.

c. Definiskan secara jelas *Game Worlds*-nya

Game worlds adalah elemen-elemen utama yang terdapat dalam suatu program game yang terdiri dari :

(i). *Game board*

Game board merupakan bentuk tampilan, latar belakang, dan lainnya.

(ii). Instruksi untuk game

Instruksi untuk game harus jelas supaya tidak membingungkan dan pemain dapat menentukan langkah dari game tersebut.

(iii). Informasi untuk pemain

Informasi ini penting ditampilkan dalam program game ketika sedang berjalan.

(iv). Penghargaan

Penghargaan digunakan sebagai rangsangan untuk pemain ketika menyelesaikan level tertentu dan mendorong untuk melanjutkan ke level berikutnya.

(v). Variasi

Variasi digunakan agar pemain tidak cepat merasa bosan tetapi variasi tidak boleh berlebihan karena dapat membuat alur cerita menjadi tidak konsisten dan membingungkan.

(vi). Tingkat Kesulitan

Tingkat kesulitan dari program game akan membuat pemain manjadi bergairah jika melewati tingkat kesulitan yang diberikan.

d. Pastikan game bisa dimainkan

Game yang tidak mungkin diselesaikan akan membuat program game tidak dapat dimainkan.

e. Rancang program sebaik mungkin

Gunakan teknik pemrograman yang sesuai dalam membuatnya dan buat program yang mudah dimodifikasi dan dikembangkan.

f. Pengujian program

Program yang selesai harus diuji. Pertama, program diuji oleh perancangnya sendiri (*alpha test*) untuk menentukan kesalahan logika yang mungkin terjadi. Kedua, program diuji oleh orang lain (*beta test*) untuk mengetahui kesalahan logika yang tidak terlihat oleh perancangnya sehingga memberi masukan, saran atau ide dalam pengembangan program game.

2.4 Permainan Halma

Halma (yang dalam bahasa Yunani berarti lompat) adalah permainan yang ditemukan tahun 1883 oleh seorang dokter bedah asal Inggris, George Howard Monks. Permainan ini dimainkan oleh tiga orang pemain di atas sebuah papan permainan berbentuk bintang. Tujuan memenangkan permainan halma adalah menyeberangkan biji-biji halma milik pemain ke tempat yang ada di seberang. Setiap pemain berjalan bergiliran dimana pemain tersebut hanya bisa menggerakkan satu biji halmanya pergiliran. Pemenang permainan halma adalah pemain yang paling cepat memindahkan semua biji halmanya ke tempat tujuan di seberang. Agar bisa berjalan dengan cepat, pemain boleh memindahkan biji halma dengan cara melompati biji yang lain, baik itu milik pemain sendiri maupun milik lawan dengan arah yang lurus. Begitu biji milik pemain melompati biji lainnya,

bila ada kesempatan melompat yang lainnya, pemain bisa melompat lagi sampai tidak ada jalan lagi [SYA05].



Gambar 2.2 Papan Permainan Halma

2.5 Pencarian Metode *Generate and Test*

Metode *Generate and Test* adalah metode yang paling sederhana dalam pencarian *heuristic*. Jika pembangkitan *possible solution* dikerjakan secara sistematis, maka prosedur akan mencari solusinya, jika ada. Tetapi jika ruang masalahnya sangat luas, mungkin memerlukan waktu yang sangat lama.

Pada prinsipnya metode ini merupakan penggabungan antara *depth-first search* dengan pelacakan mundur (*backtracking*), yaitu bergerak ke belakang menuju pada suatu keadaan awal. Nilai pengujian berupa jawaban "ya" dan "tidak" [DEW03].

Algoritma :

1. Bangkitkan suatu kemungkinan solusi. (membangkitkan suatu titik tertentu atau lintasan tertentu dari keadaan awal).
2. Uji apakah node tersebut benar-benar merupakan solusinya dengan cara membandingkan node tersebut atau node akhir dari suatu lintasan yang dipilih dengan kumpulan tujuan yang diharapkan.
3. Jika solusi ditemukan, keluar. Jika tidak ulangi kembali langkah yang pertama.

BAB III

METODOLOGI PENELITIAN

3.1 Analisis Kebutuhan

3.1.1 Metode Analisis

Metode analisis yang digunakan adalah metode analisis dengan pendekatan terstruktural (*structural approach*) yang lengkap dengan alat (tools) yang berupa jenis komputer yang dibutuhkan dan teknik (*techniques*) yaitu metode dan fungsi-fungsi yang dibutuhkan dalam pengembangan sistem sehingga hasil analisis akan menghasilkan sistem yang strukturnya dapat didefinisikan dengan baik dan jelas.

3.1.2 Hasil Analisis

Hasil analisis dari kebutuhan-kebutuhan didalam pembuatan perangkat lunak meliputi input, proses, output, serta fungsi-fungsi yang dibutuhkan dan antarmuka yang diinginkan.

a. Input

Input dari sistem ini adalah :

1. Pengaturan

Sebelum memulai permainan pengguna bisa mengatur pilihan pemain dan level komputer pada menu *pengaturan.* Pilihan pemain pada game ini adalah *pemain manusia* atau komputer. Pada *pemain komputer* tersedia pilihan tingkat kesulitan komputer yang akan menjadi lawan dalam game,

yaitu level *mudah*, *sedang* atau *sulit*. Tiap-tiap *pemain komputer* dapat diatur menggunakan tingkat kesulitan yang berbeda.

2. Input rute bagi user

   Untuk menjalankan bidak halmanya, penguna harus menginputkan rute yang dikehendaki dengan cara meng-klik titik-titik pada papan Halma. Rute tersebut harus merupakan rute yang valid agar bidak halma dapat dijalankan. Setelah rute ditentukan pengguna dapat menjalankan bidak Halmanya dengan men-klik tombol *jalan* yang tersedia.

b. Proses

Beberapa proses yang dapat dilakukan oleh sistem adalah sebagai berikut :

1. Proses awal permainan.

   Pada proses ini dengan memilih menu *game baru, game* akan menempatkan bidak pada papan permainan dengan posisi bidak berada pada posisi awal permainan. Sistem akan mengeset posisi bidak pada papan logika dan papan gui. Selain itu sistem mengeset giliran pemain.

2. Proses *pemain manusia* bermain

   Ketika *pemain manusia* mendapat giliran jalan, pemain harus menginputkan rute yang diinginkan. Rute akan dicek kevalidannya, jika rute yang dipilih adalah rute yang valid dan memenuhi persyaratan maka bidak Halma akan dipindahkan dari posisi awal ke posisi yang baru. Kemudian giliran akan diset pada giliran pemain berikutnya.

3. Proses *pemain komputer* bermain

   Ketika komputer mendapat giliran jalan, komputer akan melakukan pencarian rute optimum dari bidak-bidak miliknya menggunakan metode *generate and test*, dan menempatkan bidak yang memiliki rute paling optimal dari posisi awal ke posisi yang baru. Kemudian sistem akan mengeset giliran pada giliran pemain berikutnya.

4. Proses permainan selesai

   Proses nomer 3 dan nomer 4 akan diulang-ulang terus sampai 2 pemain telah selesai menempatkan seluruh bidak miliknya ke tempat tujuan. Kemudian peringkat kemenangan pemain akan ditampilkan pada layar.

c. Output

   Output pada sistem ini adalah status game pada suatu kondisi tertentu ketika game tersebut sedang berlangsung.

   1. Menampilkan papan halma dan bidak halma.
   2. Menampilkan perubahan posisi bidak halma dari posisi awal ke posisi baru ketika dimainkan.
   3. Menampilkan giliran pemain.
   4. Menampilkan peringkat pemenang setelah permainan selesai.
   5. Menampilkan petunjuk permainan pada status bar di bawah papan permainan.

d. Antarmuka yang diinginkan

   Suatu sistem yang memiliki antarmuka yang menarik dan informatif dengan tampilan pada saat bermain berupa *board game*.

e. Analisis kebutuhan perangkat lunak

perangkat lunak yang digunakan untuk membuat sistem adalah :

1. Sistem Operasi (Operating System)

   Windows 98, Widows 2000 atau Windows XP merupakan sistem operasi yang dapat digunakan untuk mengoperasikan sistem ini.

2. NetBeans IDE 5.5 dan jdk 1.6 sebagai editor penulisan kode dan pengembangan aplikasi

3. Adobe Photoshop dan Corel Draw 12.0 untuk merancang desain *game*.

f. Analisis kebutuhan perangkat keras

Sistem yang akan dibuat membutuhkan spesifikasi perangkat keras minimal sebagai berikut :

1. Intel Pentium II.
2. RAM 64 MB.
3. Harddisk 4 GB.
4. Monitor dengan 32 bit colour.
5. Keyboard, mouse standart.
6. CD ROM Drive 52x.

3.2 Perancangan Perangkat Lunak

3.2.1 Metode Perancangan

Metode perancangan perangkat lunak yang digunakan dalam pembuatan implementasi J2SE pada *game* Halma ini menggunakan *Unified Modeling Language* (UML). Ketika membangun model sistem dengan menggunakan UML,

tidak hanya diciptakan melalui satu model tunggal. Perlu didekati melalui sekumpulan pandangan model yang hampir independen. Karena tidak ada satu pandangan tunggal yang memadai untuk sistem kompleks. Maka ada beberapa diagram yang bersifat saling menggantikan untuk menyatakan hal yang sama dengan cara berbeda, misalnya diagram sekuen dan diagram kolaborasi. Diagram kolaborasi dan diagram sekuen sama-sama menunjukkan interaksi, namun diagram sekuen lebih mementingkan pada sekuen dari pesan yang berbasis pewaktuan sementara diagram kolaborasi pada unit yang mengirim pesan [HAR04]. Karena itu, model-model yang digunakan dalam perancangan ini adalah diagram kelas, diagram *use-case*, diagram sekuen dan diagram akitivitas. Diagram kelas digunakan untuk mendokumentasikan aspek statik dari system. Diagram *use-case*, diagram aktivitas, dan diagram sekuen digunakan untuk mendokumentasikan aspek dinamis dari sistem.

3.2.2 Desain Sistem

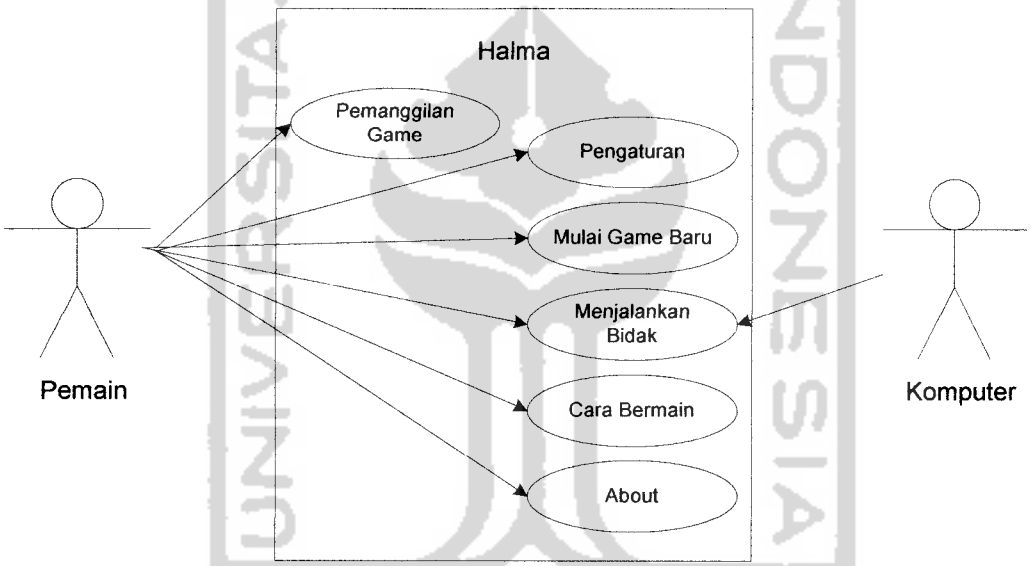
Rancang bangun sistem aplikasi *game* Halma ini terbagi dalam berbagai sudut pandang yang dapat dikategorikan ke dalam diagram berikut:

1. Use Case Diagram
2. Class Diagram
3. Diagram Sekuen
4. Activity Diagram

3.2.2.1 Diagram *Use-Case*

Diagram *use-case* digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Diagram *use-case* menyediakan cara mendeskripsikan pandangan eksternal terhadap sistem. Diagram *use-case* memiliki tiga elemen penting, yaitu actor, *use-case*, hubungan interaksi.

Dalam perancangan *game* dengan diagram *use-case* ini, digambarkan kebutuhan *game* dan interaksi antara *player* dengan *game*. Hubungan antara *game* dan *player* ditunjukkan oleh gambar 3.1.



Gambar 3.1 Diagram *use-case*

Use Case Diagram yang tejadi dalam alikasi game Halma adalah sebagai berikut :

a. Pemanggilan Game

User memanggil aplikasi game Halma untuk dijalankan.

b. Pengaturan

Pengaturan mode pemain dan tingkat kesulitan *pemain komputer*.

c. Mulai Game Baru

Memulai permainan dengan pengaturan mode pemain dan level yang sudah ditentukan sebelumnya.

d. Menjalankan Bidak

Pemain menjalankan bidak yang dimainkannya ketika giliran tiba.

e. Cara Bermain

Menberikan informasi kepada user tentang game Halma, cara bermain, dan cara mengatur mode pemain dan tingkat kesulitan komputer.

f. About

Memberikan informasi tentang versi software game dan pembuat.

3.2.2.2 Diagram Kelas

Diagram kelas digunakan untuk menunjukkan kelas-kelas yang ada di sistem dan hubungan antar kelas-kelas itu, atribut-atribut dan operasi-operasi di kelas-kelas. Diagram kelas memiliki empat elemen, yaitu kelas, antarmuka, kolaborasi, dan hubungan, seperti kebergantungan, generalisasi dan asosiasi.

Kelas-kelas yang digunakan dalam pembuatan aplikasi game Halma ini antara lain :

a. Kelas JendelaUtama

Digunakan untuk menampilkan seluruh menu dan papan permainan dalam aplikasi.

b. Kelas PapanGui

Digunakan untuk menampilkan papan permainan yang berisi bidak-bidak Halma.

c. Kelas Papan

Digunakan untuk menyimpan aturan–aturan dan pengecekan validasi pemindahan bidak pada papan permainan.

d. Kelas PapanSkor

Digunakan untuk menampilkan informasi-informasi permainan, seperti giliran, pemenanng permainan dan juga terdapat tompol untuk mengeksekusi jalannya bidak pada giliran *pemain manusia.*

e. Kelas NewBeanForm

Digunakan sebagai bidak Halma yang menampilkan gambar bidak sesuai dengan keadaan waktu permainan berjalan.

f. Kelas Pengaturan

Digunakan untuk mengatur mode pemain (manusia, komputer) dan tingkat kesulitan *pemain komputer*.

g. Kelas About

Digunakan untuk menampilkan informasi pembuat aplikasi dan versi software.

h. Kelas CaraBermain

Digunankan untuk menampilkan informasi tentang permainan Halma, cara bermain game Halma dan cara mengatur mode pemain dan mengatur tingkat kesulitan komputer.

i. Kelas Bidak

Digunakan untuk menyimpan data bidak, berupa jenis bidak, nomor bidak dan posisi bidak.

j. Kelas Lokasi

Digunakan untuk menyimpan data koordinat lokasi bidak biru, merah dan hijau pada papan permainan dan mengkonversikan koordinat lokasi bidak merah, biru dan hijau ke dalam indeks lokasi pada papan permainan maupun sebaliknya. Juga digunakan untuk menyimpan data koordinat tujuan dan koordinat larangan dimana koordinat larangan digunakan untuk mencegah masuknya bidak-bidak yang dimainkan oleh komputer pada koordinat tersebut.

k. Kelas Pemain

Digunakan membuat objek pemain, untuk menginisialisasi pemain dan menempatkan bidak-bidak pada posisi awal permainan.

l. Kelas Komputer

Digunakan untuk membuat objek *pemain komputer,* mencari rute paling optimal dan memindahkan bidak pada rute yang terpilih pada posisi awal ke posisi baru.

m. Kelas Manusia

Digunakan membuat objek *pemain manusia,* untuk memproses giliran *pemain manusia*, meliputi pengecekan validasi rute, pembatalan pilihan rute dan menyimpan rute valid yang telah dipilih.

n. Kelas ModelPemain

Digunakan untuk menyimpan variabel pengaturan model pemain.

o. Kelas ModelLevel

Digunakan untuk menyimpan variabel pengaturan level.

q. Kelas Level

Digunakan untuk mengambil data level menurut seting pengaturan.

r. Kelas LevelMudah

Digunakan untuk aturan tingkat kesulitan komputer level mudah.

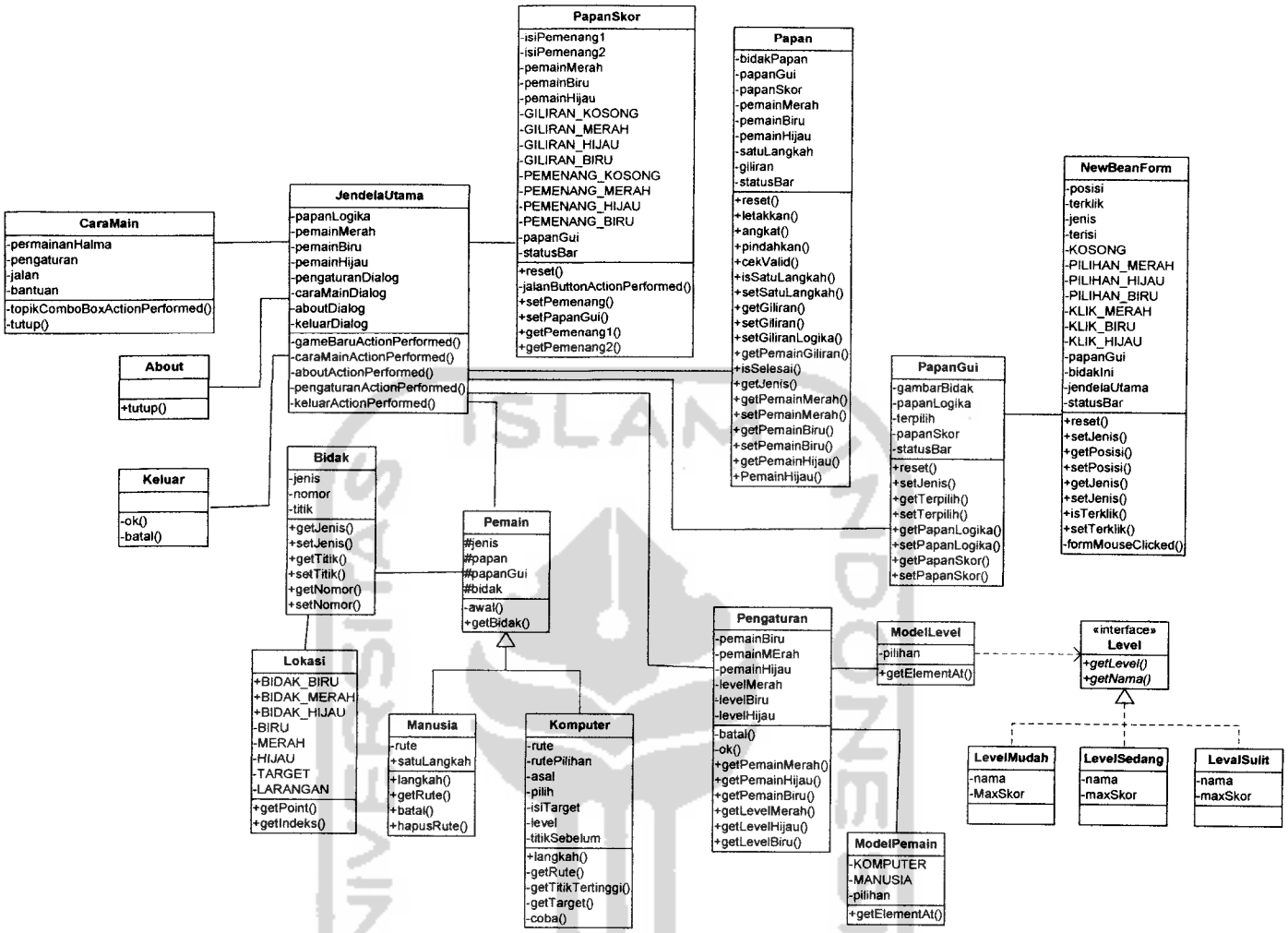
s. Kelas LevelSedang

Digunakan untuk aturan tingkat kesulitan komputer level sedang.

t. Kelas LevelSulit

Digunakan untuk aturan tingkat kesulitan komputer level sulit.

Dalam perancangan *game* Halma, diagram kelas menunjukkan hubungan antara kelas-kelas seperti yang ditunjukkan oleh gambar 3.2



Gambar 3.2 Diagram Kelas

3.2.2.3 Diagram Sekuen

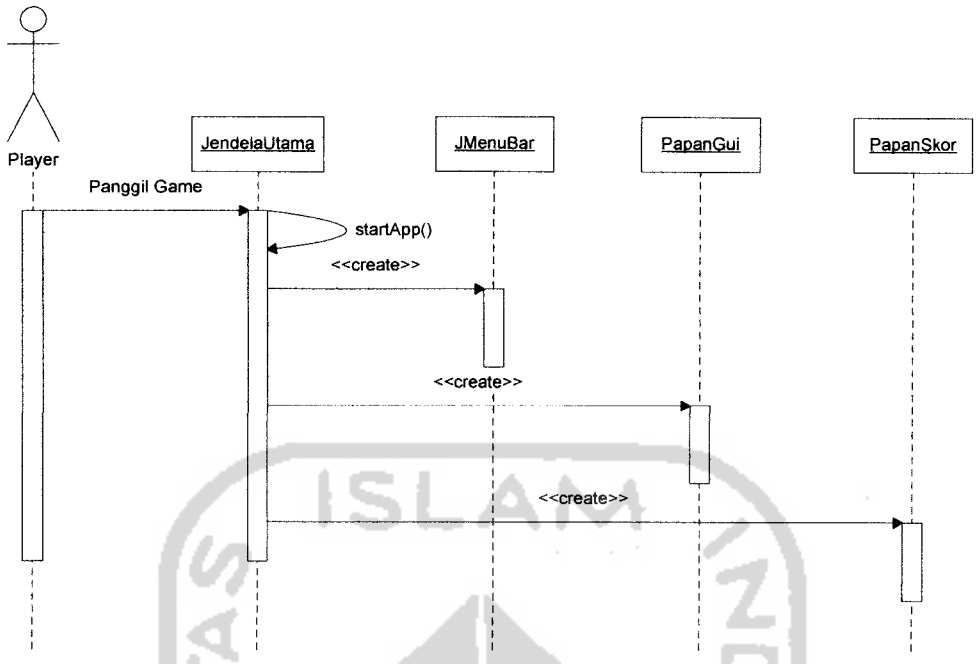
Diagram Sekuen menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dsb) berupa pesan yang digambarkan terhadap waktu. Diagram Sekuen terdiri dari dimensi vertikal (waktu) dan dimensi horisontal (objek-objek terkait). Dimensi vertikal, dari atas ke bawah menggambarkan waktu sequence dari pesan atau panggilan yang terjadi. Dan

dimensi horisontal, dari kiri ke kanan, menggambarkan tujuan objek tersebut, ke mana pesan itu dikirimkan.

Diagram Sekuen biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan output tertentu. Diawali dari men*trigger* aktivitas tersebut, proses, dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Diagram Sekuen menggambarkan perilaku sistem secara dinamis dan memperlihatkan interaksi dari objek-objek yang ada pada Pengembangan Aplikasi Puzzle Game Halma yaitu: Diagram Sekuen Pemanggilan Game, Diagram Sekuen Pengaturan, Diagram Sekuen Mulai Game Baru, Diagram Sekuen Menjalankan Bidak (manusia), Diagram Sekuen Menjalankan Bidak (komputer) Diagram Cara Bermain, dan Diagram Sekuen About.

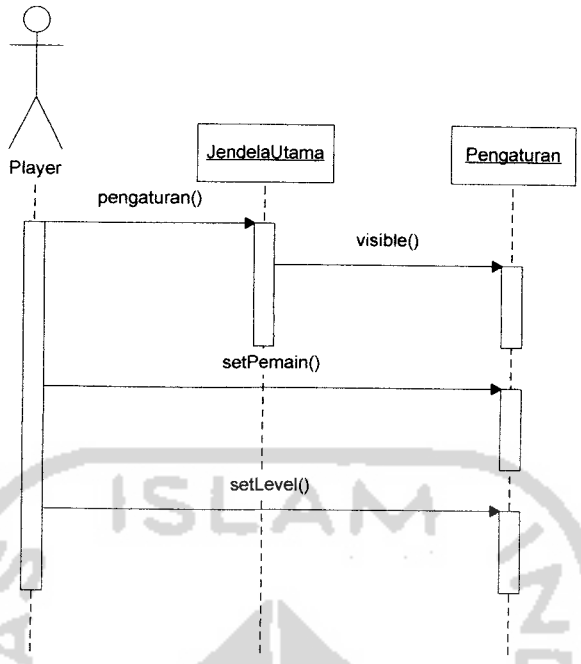
Ketika pemain menjalankan *game* pertama kali, maka kelas yang pertama kali dipanggil adalah kelas JendelaUtama. Kelas tersebut akan memunculkan menu-menu permainan dan papan permainan. Diagram sekuen pemanggilan game ditunjukkan oleh gambar 3.3.



Gambar 3.3 Diagram Sekuen pemanggilan game

Ketika pemain memilih menu *Pengaturan*, maka kelas JendelaUtama akan memanggil kelas Pengaturan. Pemain dapat memilih pilihan yang terdapat dalam form pengaturan, yaitu pilihan mode pemain dan pilihan level. Variabel yang dipilih akan disimpan.

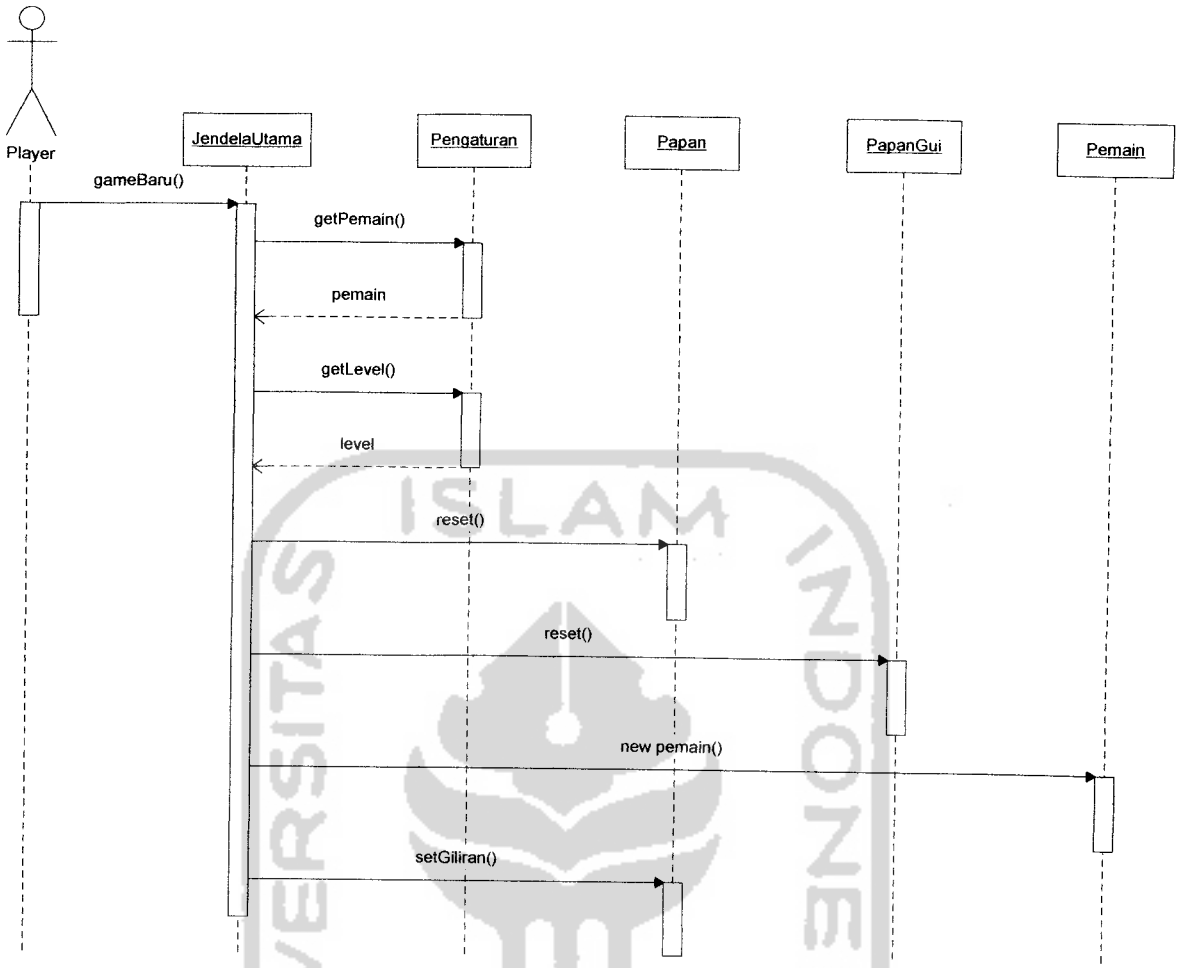
Diagram sekuen pengaturan ditunjukkan oleh gambar 3.4.



Gambar 3.4 Diagram Sekuen pengaturan

Untuk memulai permainan digunakan pilihan menu *game baru*. Ketika pemain memilih menu tersebut sistem akan mengambil varibel pengaturan mode pemain dan level komputer dan mensetting ulang papanLogika dan papanGui pada posisi awal. Sistem juga akan membuat objek *pemain* baru pada bidak sesuai dengan pilihan pengaturan.

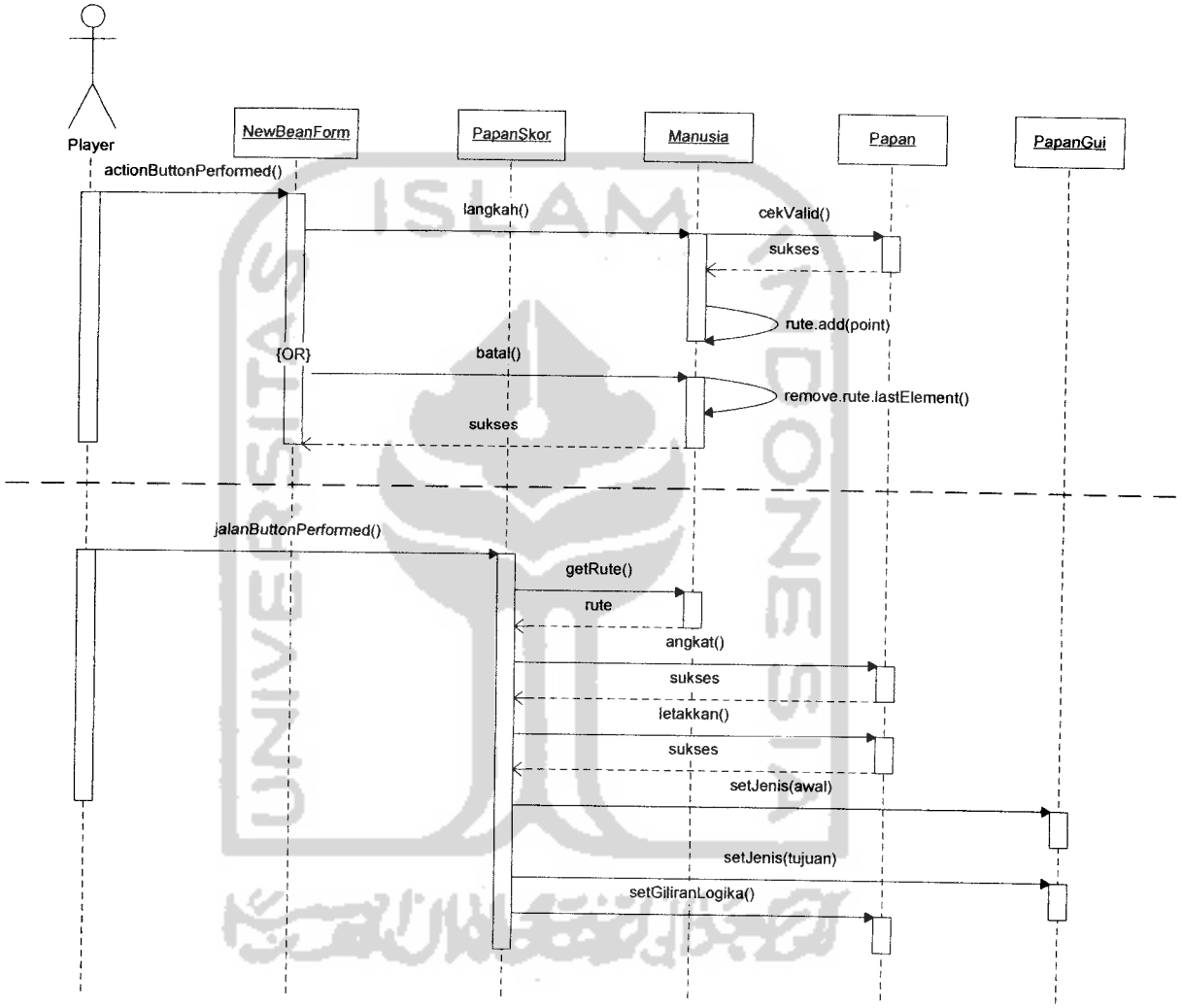
Diagram Sekuen game baru ditunjukkan oleh gambar 3.5.



Gambar 3.5 Diagram Sekuen game baru

Di saat pemain (manusia) mendapat giliran bermain, maka sistem akan menunggu perintah dari pemain melalui NewBeanForm. NewBeanForm merupakan kelas yang mewakili antarmuka bidak pada papan permainan dan dapat di klik untuk memasukkan suatu perintah. Ketika NewBenForm di beri masukan maka ia akan membaca apakah suatu masukan tersebut memenuhi syarat dan merupakan masukan untuk memilih dan menambah rute atau membatalkan rute. Setelah pemain selesai memilih rute dengan cara memberi masukan pada

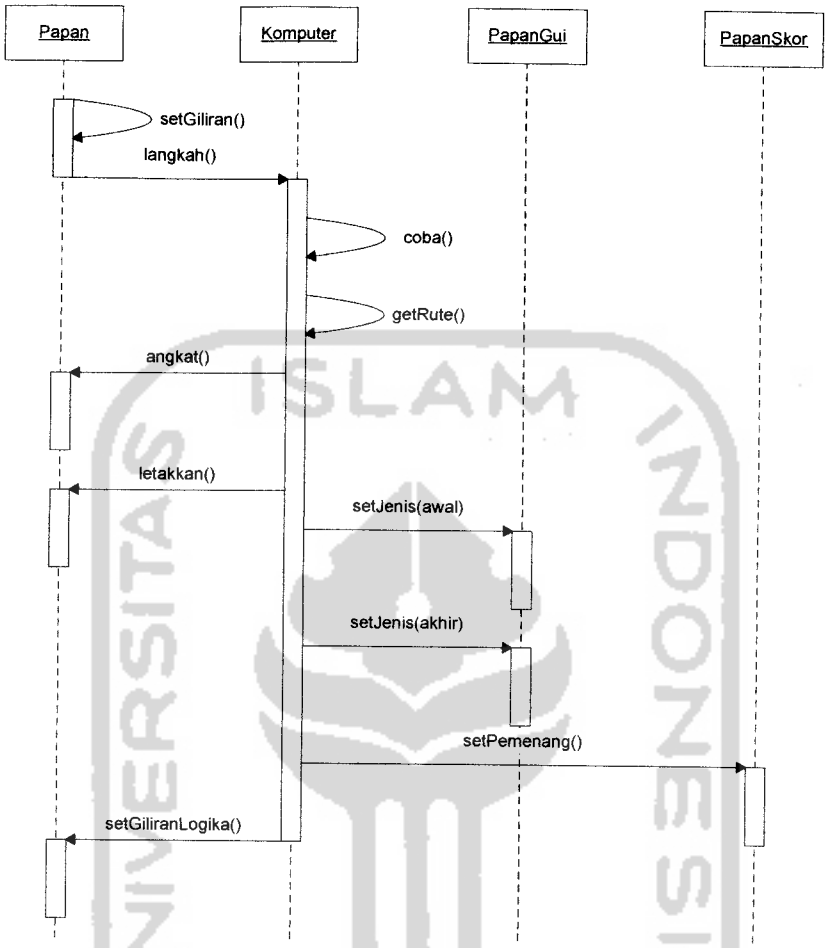
NewBeaForm, pemain dapat men-*trigger* tombol *jalan* untuk memberi perintah pada sistem agar memindahkan bidak yang terpilih dari posisi lama ke posisi baru pada rute. Sistem akan menset posisi bidak pada papanLogika dan papanGui. Diagram Sekuen Manusia Menjalankan Bidak ditunjukkan oleh gambar 3.6.



Gambar 3.6 Diagram Sekuen manusia menjalankan bidak

Pemain (komputer) bermain ketika sistem berada pada status giliran pemain yang berisi objek pemain komputer. Komputer akan mencari rute optimal bidak yang akan dijalankan dan memindahkannya dari posisi awal ke posisi baru.

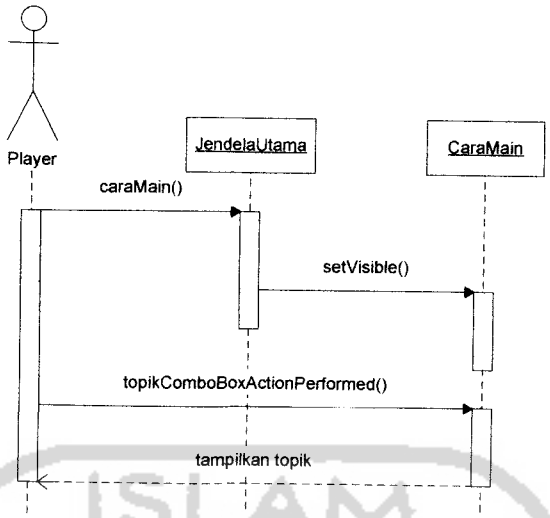
Diagram Sekuen Komputer Menjalankan Bidak ditunjukkan oleh gambar 3.7.



Gambar 3.7 Diagram Sekuen komputer menjalankan bidak

Ketika pemain memilih menu *Cara Bermain*, maka kelas JendelaUtama akan memanggil kelas CaraMain. Pemain dapat memilih pilihan yang terdapat dalam *topik box*, dan sistem akan menampilkan teks dalam teks area sesuai dengan pilihan pemain.

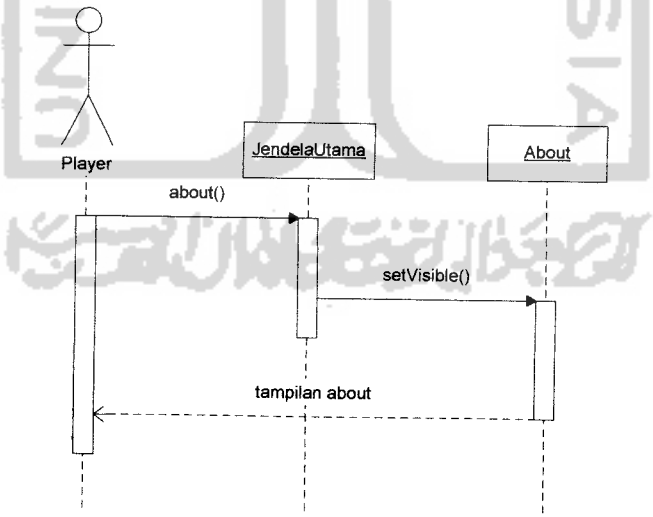
Diagram Sekuen cara bermain ditunjukkan oleh gambar 3.8.



Gambar 3.8 Diagram Sekuen cara bermain

Ketika pemain memilih menu *About*, maka kelas JendelaUtama akan memanggil kelas About. Sistem akan menampilkan informasi tentang software *game* halma.

Diagram Sekuen about ditunjukkan oleh gambar 3.9.

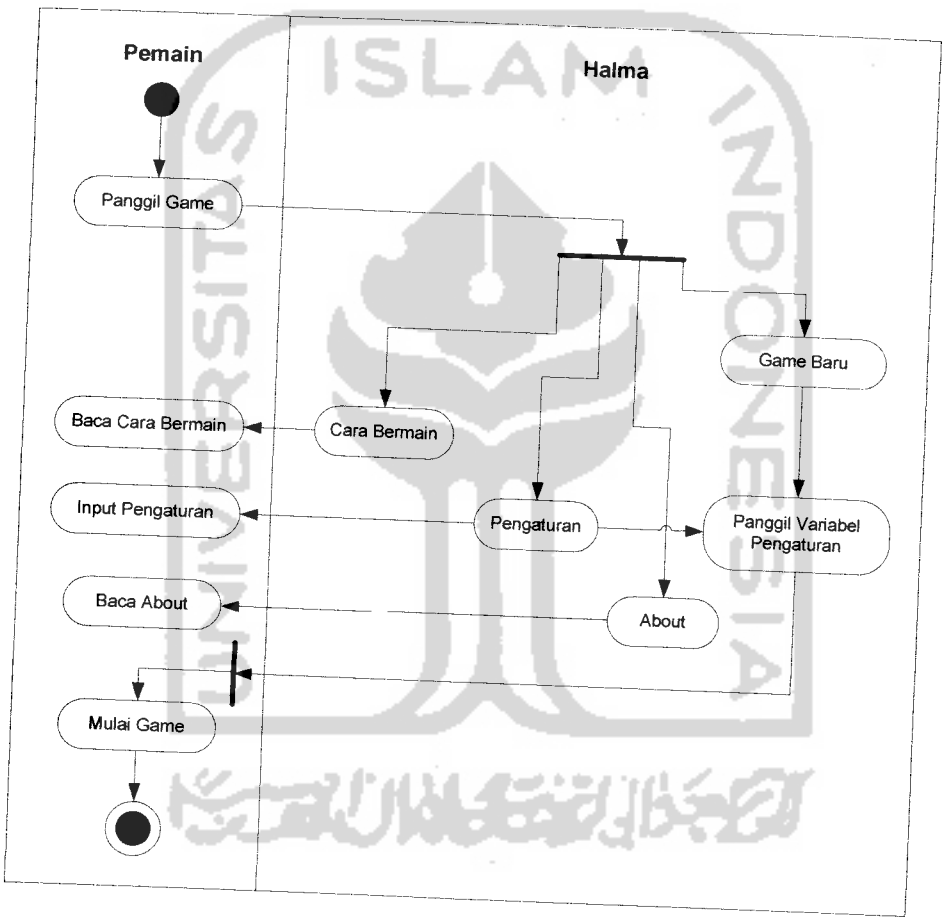


Gambar 3.9 Diagram Sekuen about

3.2.2.4 Diagram Aktifitas

Diagram aktivitas mendekripsikan aksi-aksi dan hasilnya. Diagram aktivitas memiliki tiga elemen, yaitu *activity state*, transisi, objek.

Dalam perancangan *game* Halma, diagram aktivitas digunakan untuk menggambarkan bagaimana objek-objek di dalam *game* ini bekerja. Aktivitas-aktivitas dalam *game* ini ditunjukkan oleh gambar 3.10.



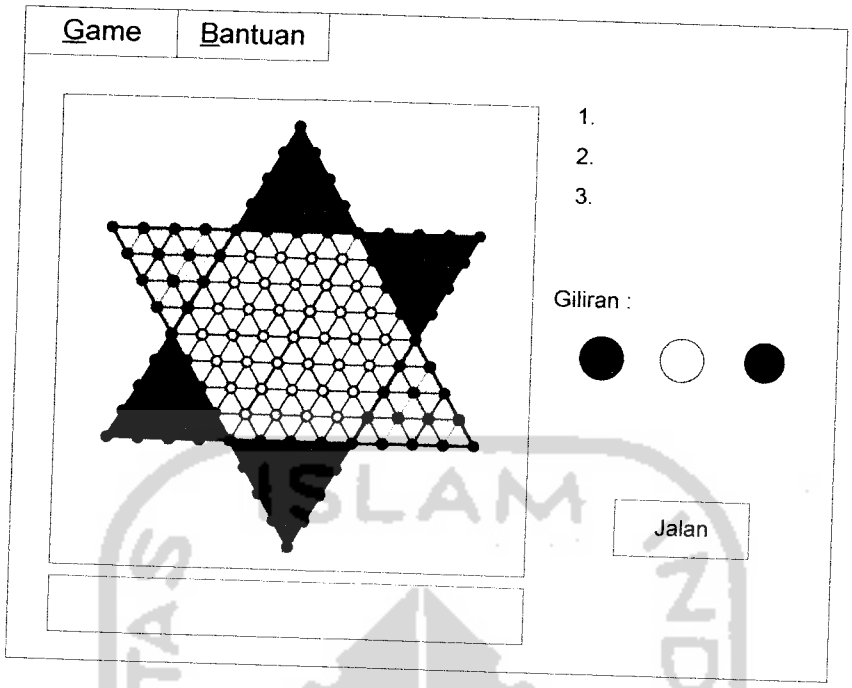
Gambar 3.10 Diagram aktivitas

3.2.3 Rancangan Antarmuka

3.2.3.1 Rancangan Antarmuka Jendela Utama

Antarmuka jendela utama mempunyai dua menu. Menu pertama yaitu *game* yang mempunyai sub menu *game baru* yang digunakan untuk memulai game baru, *pengaturan* yang digunakan untuk mengatur jenis pemain (manusia atau komputer) dan mengatur tingkat kesulitan komputer, dan *keluar* untuk keluar dari sistem. Menu kedua adalah *bantuan* yang mempunyai sub menu *cara bermain* yang isinya petunjuk cara bermain Halma dalam game ini, cara memilih pemain dan mengatur tingkat kesulitan komputer dan sejarah singkat tentang permainan Halma, dan *about* yang berisi informasi tentang data pembuat game dan tahun pembuatan. Selain terdapat dua menu tersebut juga terdapat papan permainan Halma, ikon pemain untuk menunjukkan giliran, dan tombol *jalan* untuk men-*submit* rute yang dipilih *pemain manusia* untuk diekseskusi. Terdapat juga tampilan peringkat pemenang yang muncul saat permainan berakhir. Pada bagian bawah terdapat *status bar* untuk memberi petunjuk pada *pemain manusia* saat bermain.

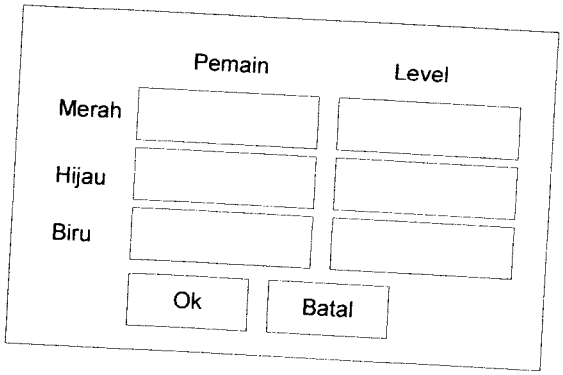
Antarmuka jendela utama seperti yang ditunjukkan oleh gambar 3.11.



Gambar 3.11 Rancangan antarmuka jendela utama

3.2.3.2 Rancangan Antarmuka Jendela Pengaturan

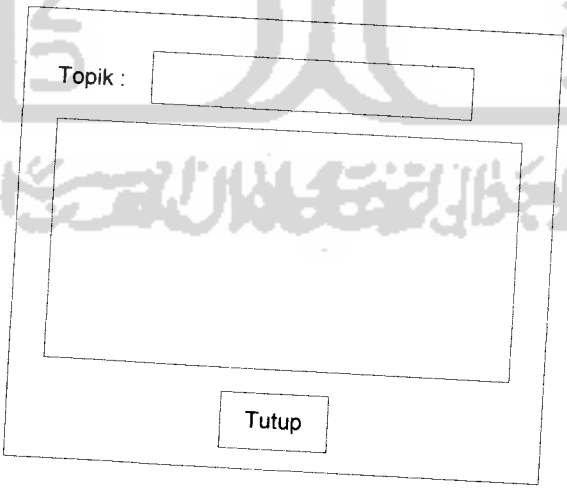
Antarmuka jendela pengaturan memiliki enam buah *combo-box*, tiga buah untuk pilihan pemain bidak merah, hijau dan biru antara *pemain manusia* dan komputer, dan tiga buah untuk pilihan tingkat kesulitan atau level komputer antara mudah, sedang dan sulit. Jika pilihan pemain pada bidak yang bersangkutan adalah manusia maka pilihan level pada bidak yang bersangkutan tidak dapat dipilih. Selain itu terdapat dua tombol yaitu tombol *ok* yang berfungsi untuk menyimpan pilihan dan keluar dari jendela pengaturan, dan tombol *batal* untuk membatalkan pilihan sekaligus keluar dari jendela pengaturan dan pilihan akan kembali seperti pilihan semula. Antarmuka jendela pengaturan seperti yang ditunjukkan oleh gambar 3.12.



Gambar 3.12 Rancangan antarmuka jendela pengaturan

3.2.3.3 Rancangan Antarmuka Jendela Cara Bermain

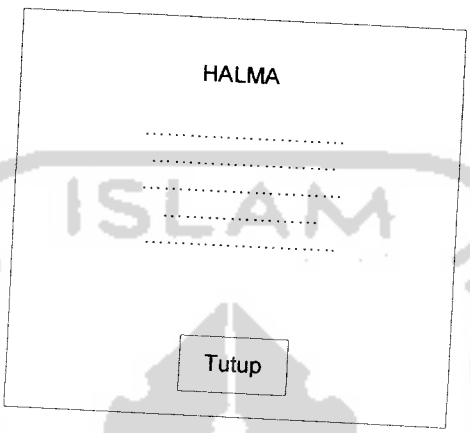
Antarmuka jendela cara bermain memiliki satu buah *combo-box* untuk memilih topik yang diinginkan. Topik yang tersedia yaitu "Permainan Halma", "Cara menjalankan bidak Halma dalam game" dan "Pengaturan level komputer". Topik yang dipilih akan ditampilkan dalam satu buah *text area-box*. Selain itu terdapat satu buah tombol *tutup* untuk keluar dari jendela cara bermain. Antarmuka jendela cara bermain seperti yang ditunjukkan oleh gambar 3.13



Gambar 3.13 Rancangan antarmuka jendela cara bermain

3.2.3.4 Rancangan Antarmuka Jendela About

Antarmuka jendela *about* menampilkan informasi tentang pembuatan game. Terdapat juga tombol *tutup* untuk keluar dari jendela *about* Antarmuka jendela about seperti yang ditunjukkan oleh gambar 3.14.



Gambar 3.14 Rancangan antarmuka jendela about

3.3 Implementasi Perangkat Lunak

Implementasi sistem merupakan tahapan yang bertujuan untuk memastikan apakah perangkat lunak yang dibuat dapat bekerja secara efektif dan efisien sesuai dengan yang diharapkan.

Sebelum program diterapkan dan diimplementasikan, maka program harus *error free* (bebas dari kesalahan). Kesalahan yang mungkin terjadi antara lain kesalahan penulisan bahasa, kesalahan sewaktu proses atau kesalahan logical. Setelah program bebas dari kesalahan, program dapat dilakukan pengujian dengan memainkan *game*.

3.3.1 Implementasi

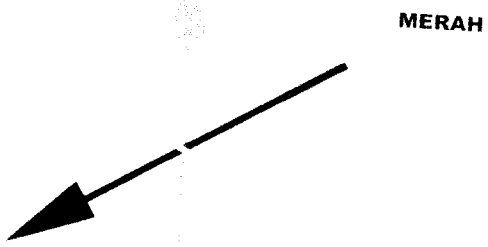
Implementasi aplikasi *game* Halma ini dibuat dengan menggunakanan bahasa pemrograman J2ME dan NetBeans 5.5 sebagai editornya. Implementasi aplikasi ini dibagi menjadi dua bagian, yaitu implementasi sistem dan implementasi antarmuka.

3.3.1.1 Implementasi Sistem

Pada bagian ini akan menjelaskan implementasi konsep game Halma pada sistem dan memuat prosedur-prosedur atau *method* yang terdapat pada aplikasi *game* Halma

3.3.1.1.1 Implementasi Konsep Permainan Halma Pada Sistem

Permainan halma adalah sebuah permainan dimana terdapat tiga pemain yang menjalankan bidaknya ke arah yang berbeda dengan tujuan yang berbeda pula di dalam satu papan permainan yang sama. Bidak-bidak milik pemain dapat beriteraksi dengan bidak milik pemain lainnya dengan cara bisa melompatinya atau menghalangi perjalanan bidak lawan. Arah perjalanan dan tujuan pemain halma ditunjukkan pada gambar 3.15.

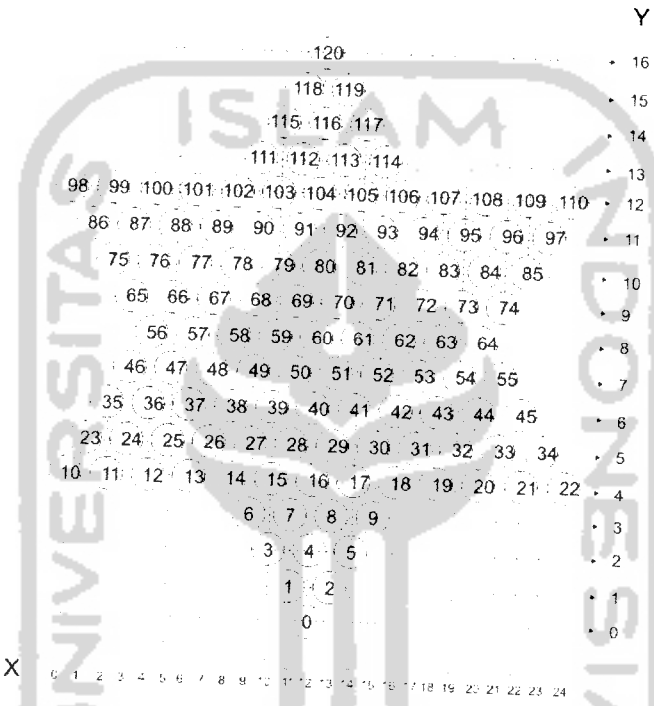


Gambar 3.15 Arah perjalanan pemain

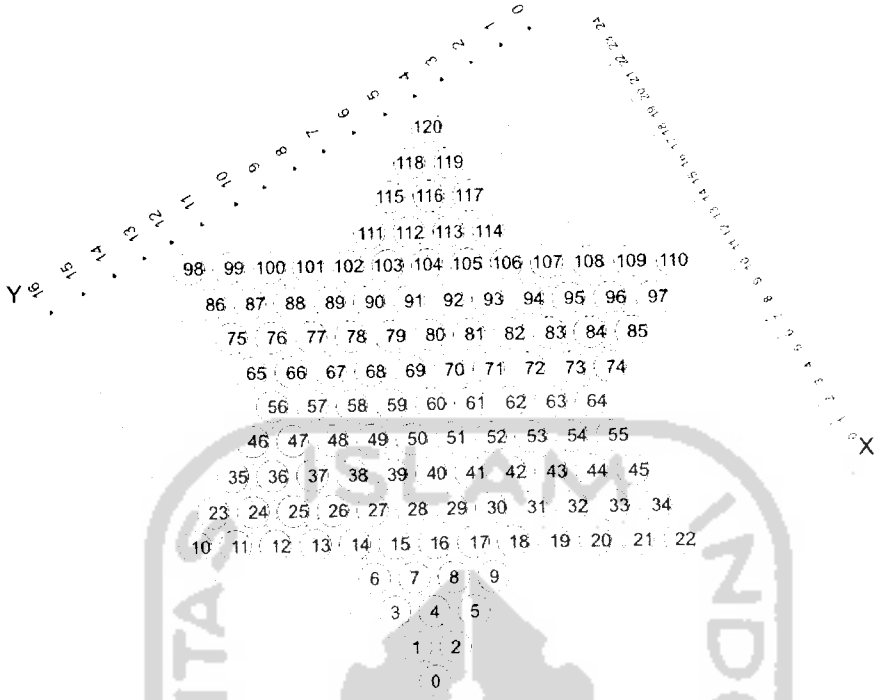
Akan tetapi sebenarnya terdapat kesamaan pada ketiga arah perjalanan tersebut yang dapat direpresentasikan dalam sistem menjadi sebuah koordinat point x dan point y yang berpasangan dengan indeks lokasi titik pada papan permainan. Koordinat masing-masing bidak merah, hijau dan biru diletakkan pada indeks lokasi papan permainan dalam posisi yang berbeda, dengan posisi awal bidak mempunyai nilai point y kecil dan tujuan bidak mempunyai nilai point y besar. Sistem dapat mengkonversikan koordinat bidak milik seorang pemain menjadi indeks lokasi pada papan permainan atau sebaliknya dapat mengkonversikan indeks lokasi pada papan permainan menjadi sebuah koordinat bagi bidak warna tertentu. Dengan cara ini pemain dapat mengetahui posisi bidak milik lawan dari sudut pandang koordinatnya. Hal ini diperlukan agar semua bidak pemain yang berbeda dapat berinteraksi dan diketahui keberadaannya oleh pemain lainnya.

Penerapan konsep ini diperlukan agar sistem dapat menerapkan satu metode yang sama dalam melakukan beberapa proses yang diperlukan untuk menjalankan permainan pada bidak-bidak yang berbeda.

Koordinat posisi masing-masing pemain ditunjukkan pada gambar 3.16, 3.17 dan 3.18.



Gambar 3.16 Koordinat posisi bidak biru



Gambar 3.17. Koordinat posisi bidak merah



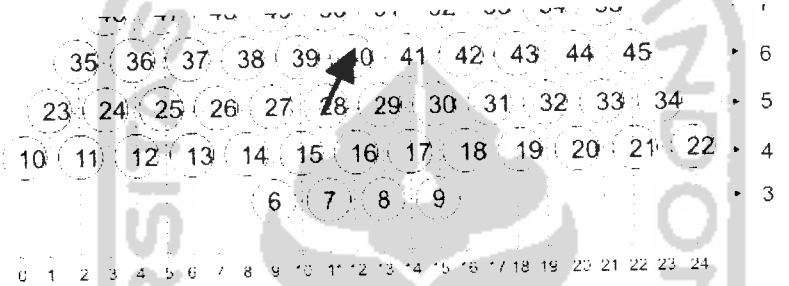
Gambar 3.18 Koordinat posisi bidak hijau

Dengan menggunakan sistem koordinat seperti di atas, langkah bidak dapat dilakukan menggunakan aturan-aturan sebagai berikut:

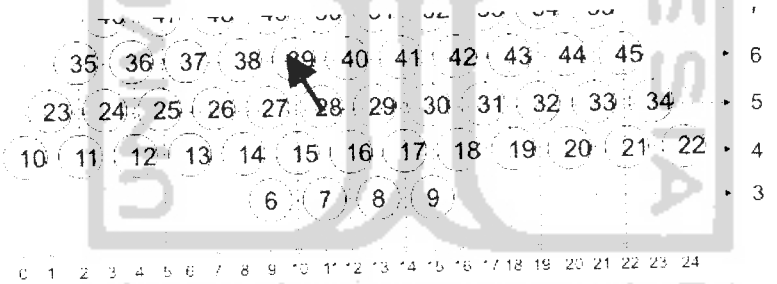
a. Aturan Bergeser

Digunakan untuk menggeser bidak. Bidak tidak dapat melangkah lagi dan hanya dapat melangkah satu langkah. Bidak bisa melangkah jika koordinat tujuan tidak terisi bidak lain milik pemain atau milik lawan.

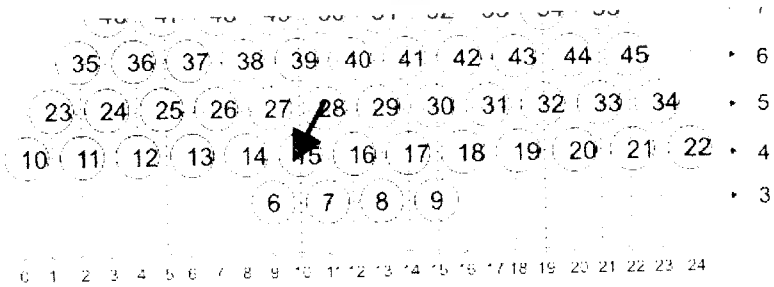
1. x+1, y+1



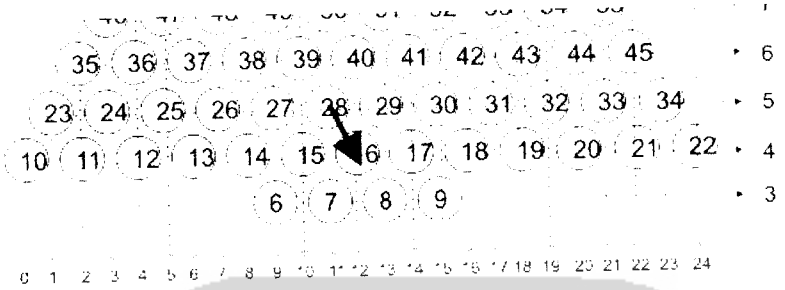
2. x-1, y+1



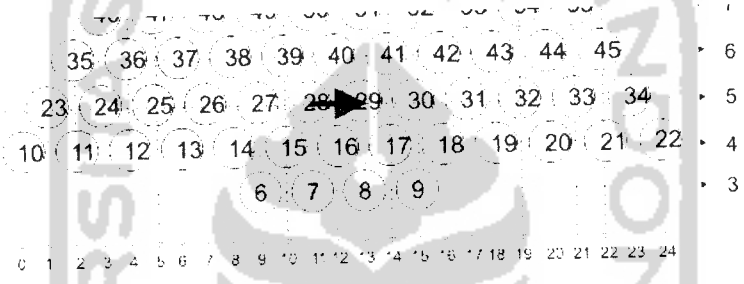
3. x-1, y-1



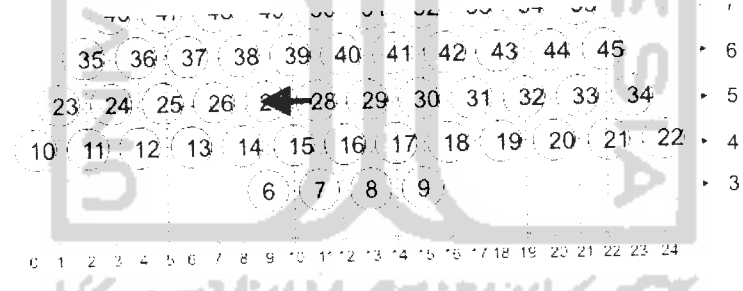
4. x+1, y-1



5. x+2, y



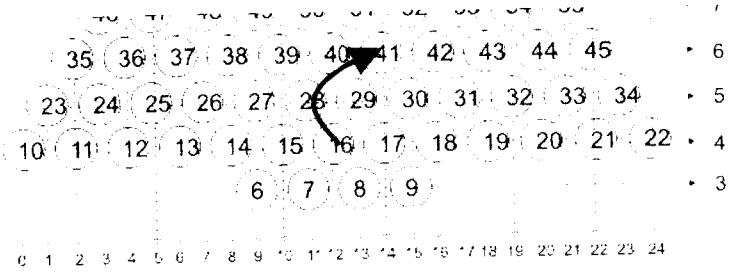
6. x-2, y



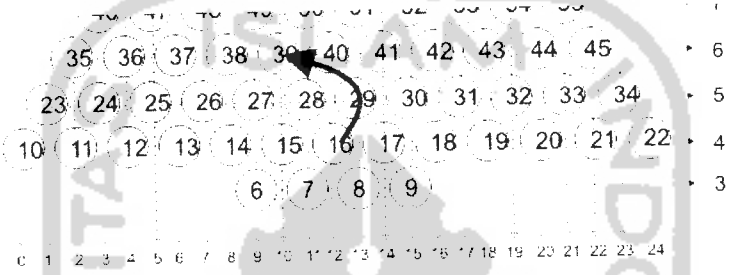
b. Aturan Lompat

Aturan ini digunakan untuk memindahkan bidak dengan cara melompati bidak lain baik milik pemain sendiri maupun milik lawan. Pada aturan dimana bidak melompati bidak lain maka akan dicek apakah ada bidak yang bisa dilompati. Contohnya pada aturan (x+2, y+2) maka koordinat (x+1, y+1) harus terisi bidak dan koordinat tujuan (x+2, y+2) harus kosong. Dimana x dan y adalah koordinat awal bidak.

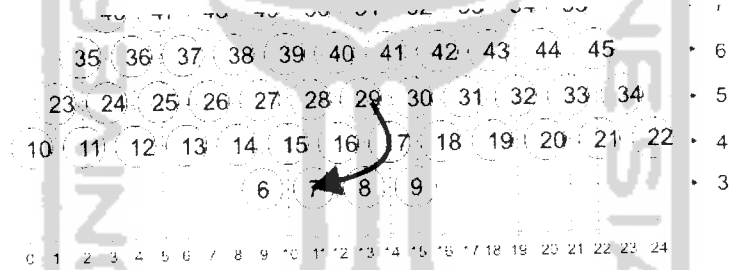
1. x+2, y+2



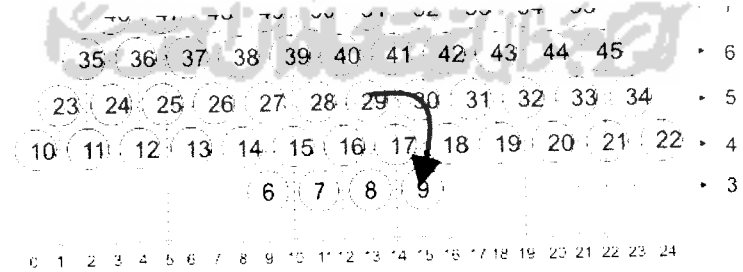
2. x-2, y+2



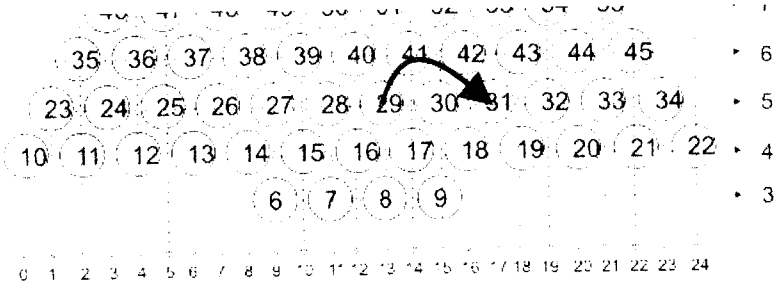
3. x-2, y-2



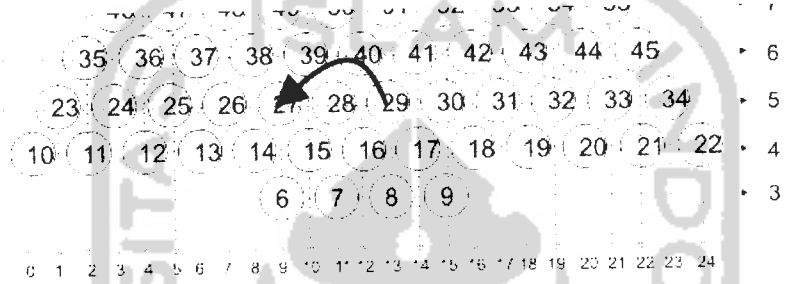
4. x+2, y-2



5. x+4, y



6. x-4, y



Pengimplementasian konsep tersebut pada sistem dapat dilihat pada sub bab berikutnya di bawah ini.

3.3.1.1.2 Penyimpanan Data Koordinat dan Konversi Koordinat - Indeks

Pada kelas Lokasi disimpan data koordinat pemain merah, hijau dan biru menggunakan *array* yang berisi data point koordinat, dimana isi koordinat *array* sesuai dengan indeks lokasi papan permainan seperti pada konsep di atas.

```
private static final Point[] BIRU = new Point [121];
private static final Point[] MERAH = new Point [121];
private static final Point[] HIJAU = new Point [121];

 * Creates a new instance of Lokasi
 */
static {
   //indeks lokasi bidak biru
   BIRU[0] = new Point(12,0);
   BIRU[1] = new Point(11,1);
   ....
   ....
```

```
....
BIRU[119] = new Point(13,15);
BIRU[120] = new Point(12,16);

//indeks lokasi bidak merah
MERAH[0] = new Point(0,12);
MERAH[1] = new Point(2,12);
....
....
....
MERAH[119] = new Point(22,4);
MERAH[120] = new Point(24,4);

//indeks lokasi bidak hijau
HIJAU[0] = new Point(24,12);
HIJAU[1] = new Point(23,11);
....
....
....
HIJAU[119] = new Point(1,5);
HIJAU[120] = new Point(0,4);
```

Untuk dapat mengkonversikan indeks lokasi pada papan permainan menjadi point koordinat maupun sebaliknya digunakan *method* yang terdapat pada kelas Lokasi sebagai berikut :

```
//mencari point dari indeks
public static Point getPoint(int bidak, int indeks) {
    Point pointTujuan = null;
    if (bidak == BIDAK_BIRU){
        pointTujuan = BIRU[indeks];
    } else if (bidak == BIDAK_MERAH){
        pointTujuan = MERAH[indeks];
    } else if (bidak == BIDAK_HIJAU){
        pointTujuan = HIJAU[indeks];
    }
    return pointTujuan;
}

//mencari indeks dari point
public static  int getIndeks(int bidak, Point point){
    int indeks = -1;
    if (bidak == BIDAK_BIRU) {
        for (int i=0; i < BIRU.length; i++){
            if (BIRU[i].x == point.x && BIRU[i].y == point.y){
                indeks = i;
                break;
            }
        }
```

```
    } else if (bidak == BIDAK_MERAH) {
        for (int i=0; i < MERAH.length; i++){
            if (MERAH[i].x == point.x && MERAH[i].y == point.y){
                indeks = i;
                break;
            }
        }
    } else if (bidak == BIDAK_HIJAU) {
        for (int i=0; i < HIJAU.length; i++){
            if (HIJAU[i].x == point.x && HIJAU[i].y == point.y){
                indeks = i;
                break;
            }
        }
    }
    return indeks;
}
```

3.3.1.1.3 Pengesetan Awal Game Baru

Untuk memulai permainan, digunakan menu *game baru*. Pada saat menu tersebut dipilih maka sistem akan menggunakan prosedur tertentu untuk menyiapkan papan permainan pada kondisi awal permainan. *Method* yang terdapat pada kelas JendelaUtama ini akan mengeset objek *papanGui*, *papanLogika* dan *papanSkor* pada kondisi awal dan membaca variabel pengaturan untuk mengeset membuat objek pemain. *Method* ini juga akan mengeset giliran dengan cara acak sehingga permainan akan dimulai oleh pemain yang terpilih secara acak.

```
public void run() {
        Random rndm = new Random();
        int bilRandom = rndm.nextInt(3);
        papanLogika = new halma.Papan(papanGui);
        papanLogika.reset()
        papanGui.reset();
        papanGui.setPapanSkor(papanSkor1);
        papanSkor1.setPapanGui(papanGui);
        papanGui.getPapanSkor().reset();
        if (pengaturanDialog.getPemainBiru() == ModelPemain.KOMPUTER) {
            pemainBiru = new Komputer(Lokasi.BIDAK_BIRU, papanLogika,
pengaturanDialog.getLevelBiru());
        } else {
            pemainBiru = new Manusia(Lokasi.BIDAK_BIRU, papanLogika);
```

```
        }
        if (pengaturanDialog.getPemainMerah() == ModelPemain.KOMPUTER) {
            pemainMerah = new Komputer(Lokasi.BIDAK_MERAH, papanLogika,
pengaturanDialog.getLevelMerah());
        } else {
            pemainMerah = new Manusia(Lokasi.BIDAK_MERAH, papanLogika);
        }
        if (pengaturanDialog.getPemainHijau() == ModelPemain.KOMPUTER) {
            pemainHijau = new Komputer(Lokasi.BIDAK_HIJAU, papanLogika,
pengaturanDialog.getLevelHijau());
        } else {
            pemainHijau = new Manusia(Lokasi.BIDAK_HIJAU, papanLogika);
        }
        papanLogika.setPemainMerah(pemainMerah);
        papanLogika.setPemainBiru(pemainBiru);
        papanLogika.setPemainHijau(pemainHijau);
        statusBar.setText("");
        System.out.println("random" +bilRandom);
        if (bilRandom == 0) {
            papanLogika.setGiliran(Lokasi.BIDAK_BIRU, statusBar);
        } else if (bilRandom == 1) {
            papanLogika.setGiliran(Lokasi.BIDAK_HIJAU, statusBar);
        } else if (bilRandom == 2) {
            papanLogika.setGiliran(Lokasi.BIDAK_MERAH, statusBar);
        }
    }
}
```

3.3.1.1.4 Pengesetan Giliran

Setiap selesai melakukan langkah dilakukan prosedur pengesetan giliran menggunakan *method setGiliranLogika()* yang terdapat pada kelas Papan. *Method* tersebut akan menentukan giliran selanjutnya dengan melihat apakah sudah terdapat pemenang pertama atau belum. Jika sudah maka pemain yang telah menang akan dilewati gilirannya.

Method setGiliranLogika akan mengeset objek giliran pada *method setGiliran* dengan variabel yang telah dipilih. Jika giliran tiba pada objek pemain yang merupakan *instance* dari kelas Komputer maka sistem akan memerintahkan *pemain komputer* untuk bermain dengan memanggil *method langkah* pada kelas Komputer.

```
//method setGiliranLogika
public void setGiliranLogika() {
      if (getGiliran() == Lokasi.BIDAK_BIRU) {
         if (papanSkor.getPemenang1() == Lokasi.BIDAK_HIJAU)
            setGiliran(Lokasi.BIDAK_MERAH, statusBar);
         else
            setGiliran(Lokasi.BIDAK_HIJAU, statusBar);
      } else if (getGiliran() == Lokasi.BIDAK_HIJAU) {
         if (papanSkor.getPemenang1() == Lokasi.BIDAK_MERAH)
            setGiliran(Lokasi.BIDAK_BIRU, statusBar);
         else
            setGiliran(Lokasi.BIDAK_MERAH, statusBar);
      } else if (getGiliran() == Lokasi.BIDAK_MERAH) {
         if (papanSkor.getPemenang1() == Lokasi.BIDAK_BIRU)
            setGiliran(Lokasi.BIDAK_HIJAU, statusBar);
         else
            setGiliran(Lokasi.BIDAK_BIRU, statusBar);
      }
   }

//method setGiliran
public void setGiliran(int giliran, JTextField statusBar) {
      this.statusBar = statusBar;
      papanSkor.setGiliranIcon(giliran);
      this.giliran = giliran;
      if (giliran == Lokasi.BIDAK_BIRU) {
         if (getPemainBiru() instanceof Komputer) {
            ((Komputer)getPemainBiru()).langkah();
         } else {
            statusBar.setText("Pilih bidak yang akan dijalankan");
            papanSkor.jalanButtonEnabled();
         }
      } else if (giliran == Lokasi.BIDAK_MERAH) {
         if (getPemainMerah() instanceof Komputer) {
            ((Komputer)getPemainMerah()).langkah();
         } else {
            statusBar.setText("Pilih bidak yang akan dijalankan");
            papanSkor.jalanButtonEnabled();
         }
      } else if (giliran == Lokasi.BIDAK_HIJAU) {
         if (getPemainHijau() instanceof Komputer) {
         ((Komputer)getPemainHijau()).langkah();
         } else {
            statusBar.setText("Pilih bidak yang akan dijalankan");
             papanSkor.jalanButtonEnabled();
         }
      }
   }
```

3.3.1.1.5 Prosedur Pemain Komputer Bermain

Pemain Komputer menggunakan *method langkah* untuk memilih bidak mana yang akan dijalankan dengan cara mencari rute yang paling optimum dan kemudian menjalankannya. Rute optimum disini adalah rute yang terdapat dalam variabel rute yang dipilih berdasarkan prioritas sebagai berikut :

1. Rute yang mempunyai skor nilai y tertinggi, yaitu nilai y yang dimiliki indeks terakhir dari element rute yang dipilih dikurangi nilai y dari indeks pertama dari element rute yang dipilih.
2. Jika terdapat dua atau lebih rute yang memiliki skor tertinggi maka dipilih rute yang mempunyai nilai y pada koordinat asalnya (indeks pertama) yang paling kecil atau paling jauh dari target.
3. Jika skor = 0 (tidak ada bidak yang dapat melangkah naik / memiliki kemajuan nilai y) maka dipilih bidak yang memiliki nilai x pada koordinat asal yang paling jauh dari nilai x dari target.

```
public boolean langkah() {
   rutePilihan = new Vector<Vector<Point>>();
   rute = new Vector<Vector<Point>>();
   pilih = new Vector<Point>();

   Point target = Lokasi.TARGET[getTarget()];

   //mencoba seluruh bidak dan menyimpannya dalam vector rute;
   for (int i = 0; i<15; i++) {
      asal = new Vector<Point>();
      coba(bidak[i],0);
      rute.addElement(getRute(asal));
      System.gc();
   }
   System.out.println("rute "+rute);
   System.out.println("rute size "+rute.size());
   int skor = 0;
   for (int i=0; i<rute.size(); i++) {
      //mencari rute dengan skor terbesar
      if (((rute.elementAt(i).lastElement().y - rute.elementAt(i).firstElement().y) > skor)
&& (rute.elementAt(i).lastElement().y - rute.elementAt(i).firstElement().y) <=
level.getMaxSkor()) {
```

```
            skor = rute.elementAt(i).lastElement().y - rute.elementAt(i).firstElement().y;
            rutePilihan.removeAllElements();
            rutePilihan.add(rute.elementAt(i));
          //jika terdapat lebih dari satu rute dengan skor terbesar maka disimpan semua
          } else if (rute.elementAt(i).size() > 1 && (rute.elementAt(i).lastElement().y -
rute.elementAt(i).firstElement().y) == skor) {
            rutePilihan.add(rute.elementAt(i));
          }
        }
        System.out.println("rutePilihan "+rutePilihan);
        System.out.println("size "+rutePilihan.size());
        int nilaiY = 16;
        //mencari rute dengan posisi awal paling jauh
        for (int i=0; i<rutePilihan.size(); i++) {
          if (skor != 0 && rutePilihan.elementAt(i).firstElement().y < nilaiY) {
            nilaiY = rutePilihan.elementAt(i).firstElement().y;
            pilih.removeAllElements();
            pilih.addAll(rutePilihan.elementAt(i));
          }
        }
        //jika skor = 0 dicari bidak dengan posisi awal y <12 (tidak pada target)
        if (skor == 0) {
          Vector<Point> calon = null;
          if (rutePilihan.size() == 0) {
            for (int i = 0; i<rute.size(); i++) {
              if (rute.elementAt(i).firstElement().y < 12) {
                calon = rute.elementAt(i);
                System.out.println("calon___"+calon);
                break;
              }
            }
          } else {
            for (int i = 0; i < rutePilihan.size(); i++) {
              if (rutePilihan.elementAt(i).firstElement().y < 12) {
                calon = rutePilihan.elementAt(i);
                break;
              }
            }
            for (int i=0; i<rutePilihan.size(); i++) {
              if (Math.abs(rutePilihan.elementAt(i).firstElement().x - target.x) >
Math.abs(calon.firstElement().x - target.x) && rutePilihan.elementAt(i).firstElement().y
< 12) {
                calon = rutePilihan.elementAt(i);
                break;
              }
            }
            System.out.println("rute nol ___"+calon);
          }
          pilih.addAll(calon);
        }
        Point awal = null;
        Point tujuan = null;
        awal = pilih.firstElement();
        if (pilih.size()<2) {
```

```
            if (papan.getJenis(Lokasi.getIndeks(papan.getGiliran(), new
        Point(pilih.elementAt(0).x+2, pilih.elementAt(0).y))) == -1) {
                tujuan = new Point(pilih.elementAt(0).x+2, pilih.elementAt(0).y);
            } else if (papan.getJenis(Lokasi.getIndeks(papan.getGiliran(), new
        Point(pilih.elementAt(0).x-2, pilih.elementAt(0).y))) == -1) {
                tujuan = new Point(pilih.elementAt(0).x-2, pilih.elementAt(0).y);
            }
        } else if (pilih.size()>1) {
            tujuan = pilih.lastElement();
        }
        Komputer komputer = (Komputer)papan.getPemainGiliran();
        Bidak bidak = komputer.getBidak(awal);
        papan.angkat(bidak);
        papan.getPapanGui().setJenis(Lokasi.getIndeks(papan.getGiliran(),awal), -1);
        papan.letakkan(bidak, tujuan);
        papan.getPapanGui().setJenis(Lokasi.getIndeks(papan.getGiliran(),tujuan),
    papan.getGiliran());
        papan.getPapanGui().getPapanSkor().setPemenang();
        papan.setGiliranLogika();
        System.gc();
    }
}
```

Dalam *method langkah* yang terdapat dalam kelas Komputer tersebut terdapat *method-method* lain yang dipanggil yaitu:

1. *Method Coba*

Method ini merupakan sebuah *method* rekursif yang akan mencoba dan menyimpan semua rute point yang mungkin dilalui bidak ke dalam variabel *asal* menggunakan ke-duabelas aturan yang telah ditentukan.

```
    private boolean coba(Bidak bidak, int aturan) {
      Point titikSkg = bidak.getTitik();    //mengambil titik koordinat bidak
      Point[] p = new Point[6];
      Point[] q = new Point[6];
      if ( aturan > p.length-1 ) {
        if (asal.size() != 0 && titikSkg.equals(asal.get(0))) { //jika vector asal sudah terisi
dan titikSkg sama dengan posisi awal bidak
          q[0] = new Point(titikSkg.x+1, titikSkg.y+1); //aturan satu langkah
          q[1] = new Point(titikSkg.x-1, titikSkg.y-1);
          q[2] = new Point(titikSkg.x+1, titikSkg.y-1);
          q[3] = new Point(titikSkg.x-1, titikSkg.y+1);
          q[4] = new Point(titikSkg.x+2, titikSkg.y);
          q[5] = new Point(titikSkg.x-2, titikSkg.y);
          for (int i = 0; i < q.length; i++) {
            if (papan.pindahkan(bidak, q[i])) { //dicoba untuk rute yang satulangkah
              asal.add(titikSkg);
```

```
                    asal.add(q[i]);
                    papan.angkat(bidak);
                    papan.letakkan(bidak,titikSkg);
                }
            }
            titikSebelum = null;
            System.gc();
            return false;
        } else { //jika tidak sama dengan posisi awal bidak dan ujicoba sampai pada
aturan ke-6
            asal.add(titikSkg);
            for (int i = 0; i < asal.size()-1; i++) {
                if (asal.get(i+1) != null && asal.get(i+1).equals(titikSkg)) { //mencari posisi
awal percabangan
                    titikSkg = asal.get(i);
                }
            }
            papan.angkat(bidak);    //memindahkan bidak ke posisi percabangan untuk
mencoba aturan lain
            papan.letakkan(bidak, titikSkg);
            boolean sukses = coba(bidak,0);
            return sukses;
        }
    }
    p[0] = new Point(titikSkg.x+2, titikSkg.y+2);  //aturan lompat
    p[1] = new Point(titikSkg.x-2, titikSkg.y-2);
    p[2] = new Point(titikSkg.x+2, titikSkg.y-2);
    p[3] = new Point(titikSkg.x-2, titikSkg.y+2);
    p[4] = new Point(titikSkg.x+4, titikSkg.y);
    p[5] = new Point(titikSkg.x-4, titikSkg.y);
    boolean sukses = false;
    boolean samaMaju = false; //sudah pernah melalui rute ini sebelumnya
    boolean samaBelakang = titikSebelum != null && titikSebelum.equals(p[aturan]);
//agar bidak tidak mundur
    boolean samaMundur = false; //sudah pernah melalui rute ini tapi sebaliknya
    for (int i = 0; i < asal.size()-1; i++) {
        if (asal.get(i+1) != null && asal.get(i+1).equals(titikSkg) &&
asal.get(i).equals(p[aturan]))
            samaMundur = true;
    }
    for (int i = 0; i<asal.size()-1; i++) {
        if (asal.get(i+1) != null && asal.get(i+1).equals(p[aturan]) &&
asal.get(i).equals(titikSkg))
            samaMaju = true;
    }
    if (!samaMundur && !samaBelakang && !samaMaju) {
        if (papan.pindahkan(bidak, p[aturan])) { //jika sukses memindahkan bidak maka
bidak dindahkan ke p[aturan] dan dicoba lagi dengan aturan pertama
            titikSebelum = titikSkg;
            asal.add(titikSkg);
            sukses = coba(bidak,0);
        } else { //jika tidak sukses maka aturan ditambah satu
            sukses = coba(bidak,aturan+1);
        }
```

```
    } else { //jika memenuhi aturan samaMundur, samaBelakang dan samaMaju maka
aturan ditambah satu
        sukses = coba(bidak,aturan+1);
    }
    return sukses;
  }
```

2. *Method getRute*

Method getRute digunakan untuk memasukkan semua elemen variabel asal sampai dengan titik tertinggi ke dalam variabel rute.

```
  private Vector<Point> getRute(Vector<Point> asal){
    Vector<Point> calon = new Vector<Point>();
    for (Point p : asal){
      calon.add(p);
      if(p.equals(getTitikTertinggi(asal))) {
        break;
      }
    }
    return calon;
  }
```

3. *Method getTitikTertinggi*

Method GetTitikTertinggi() digunakan untuk mencari titik tertinggi dimana titik tertinggi yang dimaksud adalah koordinat yang pernah dicoba dan disimpan dalam variabel asal yang memiliki point y paling tinggi. Jika terdapat lebih dari satu koordinat dalam variabel asal yang memiliki point y tertinggi maka dipilih dari koordinat-koordinat tersebut yang memiliki point x paling dekat dengan point x target. Akan tetapi koordinat tersebut tidak boleh sama dengan koordinat *larangan* dimana bidak milik *pemain komputer* tidak diperbolehkan menempati lokasi tersebut untuk tujuan mengoptimalkan langkah selanjutnya.

```
private Point getTitikTertinggi(Vector<Point> asal){
    Point calon = asal.get(0);
    Point target = Lokasi.TARGET[getTarget()];
    for (int i = 1; i < asal.size(); i++) {
      boolean larangan = false;
```

```
            for (int n = 0; n < Lokasi.LARANGAN.length; n++) {
                if (asal.get(i).equals(Lokasi.LARANGAN[n])) {
                    larangan = true;
                    break;
                }
            }
            if (!larangan && asal.get(i).y > calon.y) {
                calon = asal.get(i);
            } else if (!larangan && asal.get(i).y == calon.y && Math.abs(asal.get(i).x - target.x)
< Math.abs(calon.x-target.x)) {
                calon = asal.get(i);
            }
        }
        return calon;
    }
```

4. *Method getTarget*

Method ini digunakan untuk mencari target yang telah di definisikan dalam kelas Lokasi yang belum ditempati bidak milik pemain.

```
    public int getTarget() {
        int target = -1;
        for (int i = 0; i<isiTarget.length; i++) {
            if (papan.getJenis(Lokasi.getIndeks(papan.getGiliran(),Lokasi.TARGET[i])) !=
papan.getGiliran()) {
                target = i;
                break;
            }
        }
        return target;
    }
```

3.3.1.1.6 Prosedur Pemain Manusia Bermain

Pemain manusia menggunakan dua *method* pada kelas Manusia yaitu *method langkah* untuk menyimpan rute yang telah dipilih dengan men-klik bidak dan rute yang terdapat pada papan permainan dan *method batal* untuk membatalkan rute yang telah dipilih. Rute yang dibatalkan harus merupakan rute yang terakhir dipilih, sehingga untuk membatalkan beberapa rute harus berurutan mulai dari rute yang paling akhir dipilih.

```
//method langkah
public boolean langkah(Point point) {
      boolean sukses = false;

      //jika belum terdapat bidak yang dipilih
      if (rute.size() == 0) {
         rute.add(point);
         sukses = true;

      //jika rute yang dimasukkan sudah terdapat didalam variabel rute
      } if (rute.contains(point)) {
         sukses = false;

     //jika sudah terdapat bidak yang dipilih dan belum ada rute yang dimasukkan
      } else if (rute.size() == 1) {
         sukses = papan.cekValid(papan.getGiliran(), rute.lastElement(), point);
         if (sukses && papan.isSatuLangkah()) {
            rute.add(point);
            satuLangkah = true;
         } else if (sukses && !papan.isSatuLangkah()) {
            rute.add(point);
            satuLangkah = false;
         }

     //jika akan memasukkan rute selanjutnya maka dicek apakah langkah sebelumnya
merupakan satu langkah atau tidak
      } else if (rute.size() > 1 && satuLangkah) {
         return false;
      } else if (rute.size() > 1 && !satuLangkah) {
         sukses = (papan.cekValid(papan.getGiliran(), rute.lastElement(), point) &&
!papan.isSatuLangkah());
         if (sukses)
            rute.add(point);
      }
      return sukses;
   }

//method batal
public boolean batal(Point point) {
      boolean sukses = false;
      if (rute != null && rute.size() > 0 && rute.lastElement().equals(point)) {
         sukses = true;
         rute.remove(rute.lastElement());
      }
      return sukses;
   }
```

Untuk menjalankan bidak yang telah dipilih *pemain manusia* digunakan tombol *jalan* yang akan men-*trigger method* pada kelas PapanSkor sebagai berikut :

```
public void run() {
        Manusia manusia = (Manusia)papanLogika.getPemainGiliran();
        if (getPemenang1() != -1 && getPemenang2() != -1) {
           return;
        } else if (manusia.getRute().size() < 2) {
           return;
        } else {
           statusBar.setText("");
           Bidak bidak = manusia.getBidak(manusia.getRute().firstElement());
           int awal = Lokasi.getIndeks(papanLogika.getGiliran(),
manusia.getRute().firstElement());
           int akhir = Lokasi.getIndeks(papanLogika.getGiliran(),
manusia.getRute().lastElement());
           for (int i = 0; i < manusia.getRute().size(); i++) {
              int kosong = Lokasi.getIndeks(papanLogika.getGiliran(),
manusia.getRute().elementAt(i));
              System.out.println("element"+kosong);
              papanLogika.getPapanGui().setJenis(kosong, -1);
           }
           papanLogika.angkat(bidak);
           papanLogika.getPapanGui().setJenis(awal, -1);
           papanLogika.letakkan(bidak, manusia.getRute().lastElement());
           papanLogika.getPapanGui().setJenis(akhir, papanLogika.getGiliran());
           setPemenang();
           manusia.hapusRute();
           papanLogika.getPapanGui().setTerpilih(null);
           jalanButtonDisabled();
           papanLogika.setGiliranLogika();
        }
     }
  }
```

3.3.1.1.7 *Method cekValid*

Method pada kelas Papan ini adalah prosedur yang berfungsi untuk mengecek apakah rute yang dipilih oleh *pemain manusia* atau *pemain komputer* adalah rute yang valid atau dapat dijalankan. Prosedur ini menggunakan ke-12 aturan untuk mengecek valid atau tidaknya rute yang telah dipilih berdasarkan perhitungan koordinat posisi yang dimiliki pemain.

```
public boolean cekValid(int bidak, Point asal, Point tujuan) {
     boolean sukses = false;
     if (asal.x+2 == tujuan.x && asal.y == tujuan.y) {
        int indeks = Lokasi.getIndeks(bidak, new Point(asal.x+2, asal.y));
        if (indeks > -1 && bidakPapan[indeks] == -1) {
           sukses = true;
           satuLangkah = true;
        }
```

```
} else if (asal.x-2 == tujuan.x && asal.y == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x-2, asal.y));
    if (indeks > -1 && bidakPapan[indeks] == -1) {
        sukses = true;
        satuLangkah = true;
    }
} else if (asal.x+1 == tujuan.x && asal.y+1 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x+1, asal.y+1));
    if (indeks > -1 && bidakPapan[indeks] == -1) {
        sukses = true;
        satuLangkah = true;
    }
} else if (asal.x-1 == tujuan.x && asal.y+1 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x-1, asal.y+1));
    if (indeks > -1 && bidakPapan[indeks] == -1) {
        sukses = true;
        satuLangkah = true;
    }
} else if (asal.x-1 == tujuan.x && asal.y-1 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x-1, asal.y-1));
    if (indeks > -1 && bidakPapan[indeks] == -1) {
        sukses = true;
        satuLangkah = true;
    }
} else if (asal.x+1 == tujuan.x && asal.y-1 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x+1, asal.y-1));
    if (indeks > -1 && bidakPapan[indeks] == -1) {
        sukses = true;
        satuLangkah = true;
    }
} else if (asal.x+2 == tujuan.x && asal.y+2 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x+1, asal.y+1));
    int indeks2 = Lokasi.getIndeks(bidak, new Point(asal.x+2, asal.y+2));
    if (indeks2 > -1 && bidakPapan[indeks2] == -1 && bidakPapan[indeks] != -1) {
        sukses = true;
        satuLangkah = false;
    }
} else if (asal.x-2 == tujuan.x && asal.y-2 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x-1, asal.y-1));
    int indeks2 = Lokasi.getIndeks(bidak, new Point(asal.x-2, asal.y-2));
    if (indeks2 > -1 && bidakPapan[indeks2] == -1 && bidakPapan[indeks] != -1) {
        sukses = true;
        satuLangkah = false;
    }
} else if (asal.x+2 == tujuan.x && asal.y-2 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x+1, asal.y-1));
    int indeks2 = Lokasi.getIndeks(bidak, new Point(asal.x+2, asal.y-2));
    if (indeks2 > -1 && bidakPapan[indeks2] == -1 && bidakPapan[indeks] != -1) {
        sukses = true;
        satuLangkah = false;
    }
} else if (asal.x-2 == tujuan.x && asal.y+2 == tujuan.y) {
    int indeks = Lokasi.getIndeks(bidak, new Point(asal.x-1, asal.y+1));
    int indeks2 = Lokasi.getIndeks(bidak, new Point(asal.x-2, asal.y+2));
    if (indeks2 > -1 && bidakPapan[indeks2] == -1 && bidakPapan[indeks] != -1) {
```

```
            sukses = true;
            satuLangkah = false;
        }
    } else if (asal.x+4 == tujuan.x && asal.y == tujuan.y) {
        int indeks = Lokasi.getIndeks(bidak, new Point(asal.x+2, asal.y));
        int indeks2 = Lokasi.getIndeks(bidak, new Point(asal.x+4, asal.y));
        if (indeks2 > -1 && bidakPapan[indeks2] == -1 && bidakPapan[indeks] != -1) {
            sukses = true;
            satuLangkah = false;
        }
    } else if (asal.x-4 == tujuan.x && asal.y == tujuan.y) {
        int indeks = Lokasi.getIndeks(bidak, new Point(asal.x-2, asal.y));
        int indeks2 = Lokasi.getIndeks(bidak, new Point(asal.x-4, asal.y));
        if (indeks2 > -1 && bidakPapan[indeks2] == -1 && bidakPapan[indeks] != -1) {
            sukses = true;
            satuLangkah = false;
        }
    }
    return sukses;
}
```

3.3.1.1.8 Pemindahan Bidak

Ketika memindahkan bidak sistem akan mengeset indeks posisi lama pada papan menjadi kosong dan mengeset indeks posisi baru pada papan menjadi bidak milik pemain. Sistem juga mengeset variabel *titik* pada objek bidak dengan koordinat posisi yang baru.

Untuk memindahkan bidak dari posisi awal ke posisi baru digunakan *method-method* pada kelas Papan sebagai berikut :

```
private boolean letakkan(int bidak, int indeks) {
    boolean sukses = false;
    System.out.println("indeks "+indeks);
    if (bidakPapan[indeks] == -1) { //jika tidak ada isinya
        bidakPapan[indeks] = bidak;
        sukses = true;
    }
    return sukses;
}

public boolean letakkan(Bidak bidak) {
    return letakkan(bidak.getJenis(), bidak.getTitik());
}

public boolean letakkan(Bidak bidak, Point point) {
    bidak.setTitik(point);
```

```
    return letakkan(bidak.getJenis(), point);
}

public boolean letakkan(int bidak, Point point) {
    int indeks = Lokasi.getIndeks(bidak, point);
    return letakkan(bidak, indeks);
}

public boolean angkat(Bidak bidak) {
    return angkat(bidak.getJenis(), bidak.getTitik());
}

public boolean angkat(int bidak, Point point){
    boolean sukses = false;
    int indeks = Lokasi.getIndeks(bidak, point);
    if (bidakPapan[indeks] != -1) { //jika ada isinya
        bidakPapan[indeks] = -1;
        sukses = true;
    }
    return sukses;
}

public boolean pindahkan(Bidak bidak, Point tujuan) {
    boolean sukses = pindahkan(bidak.getJenis(), bidak.getTitik(), tujuan);
    if (sukses) {
        bidak.setTitik(tujuan);
    }
    return sukses;
}

public boolean pindahkan(int bidak, Point asal, Point tujuan) {
    boolean sukses = false;
    sukses = angkat(bidak, asal);
    if (sukses) {
        sukses = cekValid(bidak, asal, tujuan);
    }
    if (sukses) {
        sukses = letakkan(bidak, tujuan);
    }
    if (!sukses) {
        int indeks = Lokasi.getIndeks(bidak, asal);
        bidakPapan[indeks] = bidak;
    }
    return sukses;
}
```

3.3.1.1.9 Pengecekan dan Pengesetan Pemenang

Setiap selesai melakukan langkah dilakukan pengecekan apakah semua bidak telah sampai pada target. Prosedurnya adalah sebagai berikut :

```
public boolean isSelesai(int jenis) {
    for (int i = 0; i<Lokasi.TARGET.length; i++) {
        if (bidakPapan[Lokasi.getIndeks(jenis, Lokasi.TARGET[i])] != jenis) {
            return false;
        }
    }
    return true;
}
```

Kemudian dilakukan pengesetan pemenang menurut pada parameter yang dikembalikan pada *method* pengecekan.

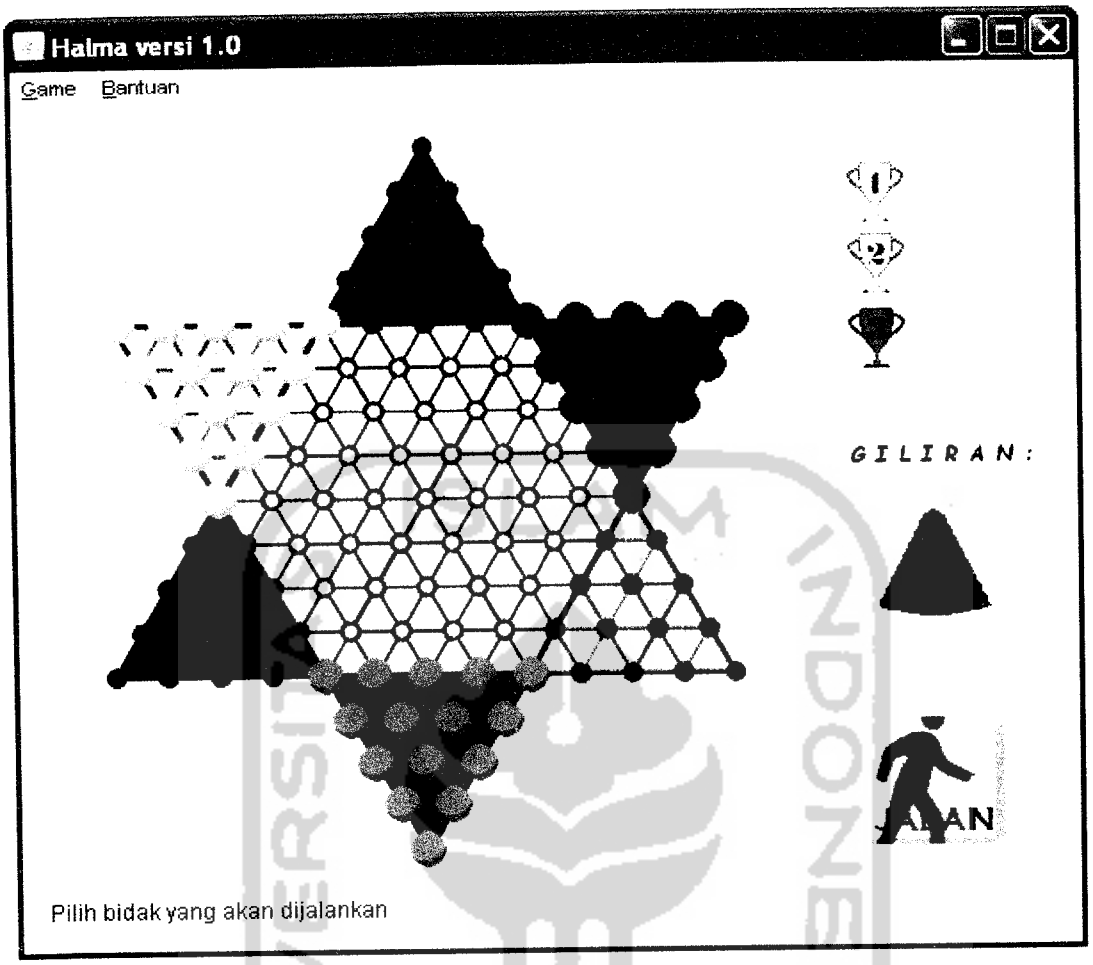
```
public void setPemenang() {
    if (!papanLogika.isSelesai(papanLogika.getGiliran())) {
        return;
    } else if (papanLogika.isSelesai(papanLogika.getGiliran()) && isiPemenang1 == -1) {
        if (papanLogika.getGiliran() == Lokasi.BIDAK_BIRU)
            pemenang1.setIcon(PEMENANG_BIRU);
        else if (papanLogika.getGiliran() == Lokasi.BIDAK_MERAH)
            pemenang1.setIcon(PEMENANG_MERAH);
        else if (papanLogika.getGiliran() == Lokasi.BIDAK_HIJAU)
            pemenang1.setIcon(PEMENANG_HIJAU);
        isiPemenang1 = papanLogika.getGiliran();
    } else if (papanLogika.isSelesai(papanLogika.getGiliran()) && isiPemenang2 == -1) {
        if (papanLogika.getGiliran() == Lokasi.BIDAK_BIRU)
            pemenang2.setIcon(PEMENANG_BIRU);
        else if (papanLogika.getGiliran() == Lokasi.BIDAK_MERAH)
            pemenang2.setIcon(PEMENANG_MERAH);
        else if (papanLogika.getGiliran() == Lokasi.BIDAK_HIJAU)
            pemenang2.setIcon(PEMENANG_HIJAU);
        isiPemenang2 = papanLogika.getGiliran();
        if ((isiPemenang1 == Lokasi.BIDAK_BIRU || isiPemenang1 ==
Lokasi.BIDAK_MERAH) && (isiPemenang2 == Lokasi.BIDAK_BIRU || isiPemenang2 ==
Lokasi.BIDAK_MERAH))
            pemenang3.setIcon(PEMENANG_HIJAU);
        else if ((isiPemenang1 == Lokasi.BIDAK_HIJAU || isiPemenang1 ==
Lokasi.BIDAK_MERAH) && (isiPemenang2 == Lokasi.BIDAK_HIJAU || isiPemenang2 ==
Lokasi.BIDAK_MERAH))
            pemenang3.setIcon(PEMENANG_BIRU);
        else if ((isiPemenang1 == Lokasi.BIDAK_HIJAU || isiPemenang1 ==
Lokasi.BIDAK_BIRU) && (isiPemenang2 == Lokasi.BIDAK_HIJAU || isiPemenang2 ==
Lokasi.BIDAK_BIRU))
            pemenang3.setIcon(PEMENANG_MERAH);
        papanLogika.setGiliran(-1);
        setTombolJalan(0);
    }
```

3.3.1.2 Implementasi Antarmuka

Pada bagian ini memuat gambaran penjelasan antarmuka yang terdapat pada sistem aplikasi *game* Halma.

3.3.1.2.1 Antarmuka Jendela Utama

Antarmuka jendela utama adalah antarmuka utama aplikasi *game* Halma dimana permainan akan berjalan pada form tersebut. Dalam form ini terdapat dua menu utama, yang pertama yaitu menu *Game* yang terdiri dari menu *Game Baru* untuk memulai permainan baru, menu *Pengaturan* untuk mengeset tipe pemain dan tingkat kesulitan *pemain komputer* dan menu *Keluar* untuk keluar dari aplikasi. Selain itu terdapat papan permainan Halma, tampilan peringkat pemenang, tampilan giliran pemain yang berjalan, petunjuk bermain untuk *pemain manusia* dan tombol *jalan* untuk mengeksekusi rute yang telah dipilih. Antarmuka jendela utama ditunjukkan pada gambar 3.19.

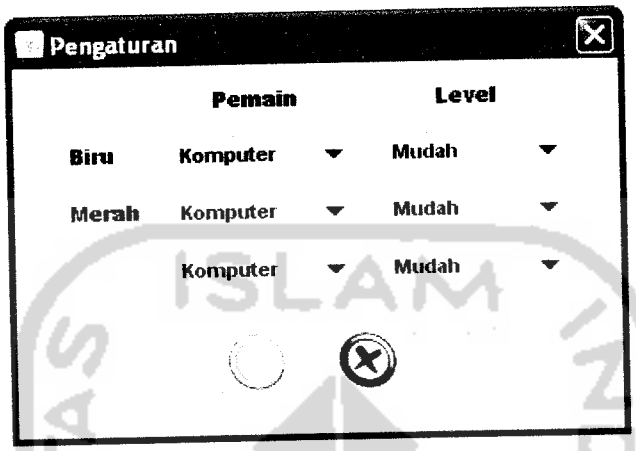


Gambar 3.19. Tampilan antarmuka jendela utama

3.3.1.2.2 Antarmuka Jendela Pengaturan

Antarmuka jendela pengaturan memiliki enam buah *combo-box*, tiga buah untuk pilihan pemain bidak merah, hijau dan biru antara *pemain manusia* dan komputer, dan tiga buah untuk pilihan tingkat kesulitan atau level komputer antara mudah, sedang dan sulit. Jika pilihan pemain pada bidak yang bersangkutan adalah manusia maka pilihan level pada bidak yang bersangkutan tidak dapat dipilih. Selain itu terdapat dua tombol yaitu tombol *ok* yang berfungsi untuk menyimpan pilihan dan keluar dari jendela pengaturan, dan tombol *batal* untuk

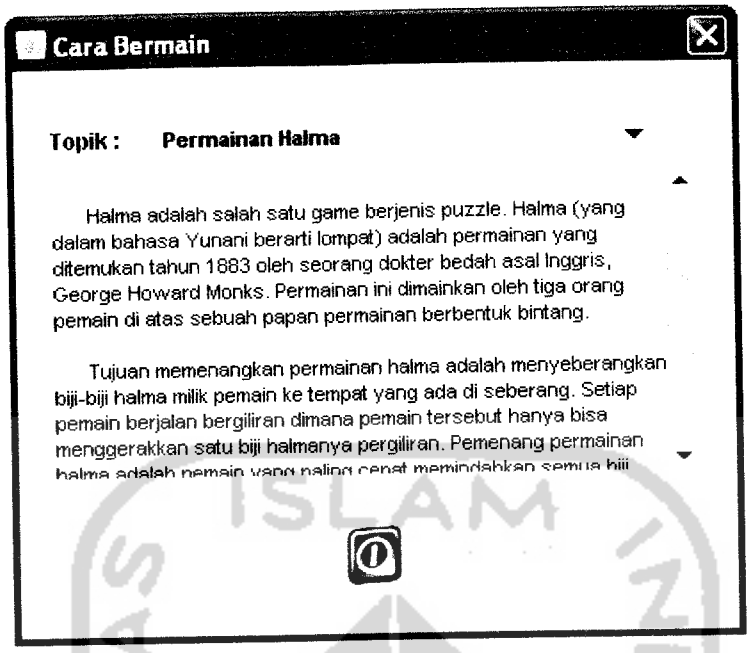
membatalkan pilihan sekaligus keluar dari jendela pengaturan dan pilihan akan kembali seperti pilihan semula. Antarmuka jendela pengaturan ditunjukkan pada gambar 3.20.



Gambar 3.20 Tampilan antarmuka jendela pengaturan

3.3.1.2.3 Antarmuka Jendela Cara Bermain

Antarmuka jendela cara bermain memiliki satu buah *combo-box* untuk memilih topik yang diinginkan. Topik yang tersedia yaitu "Permainan Halma", "Cara menjalankan bidak Halma dalam game" dan "Pengaturan level komputer". Topik yang dipilih akan ditampilkan dalam satu buah *text area-box*. Selain itu terdapat satu buah tombol *tutup* untuk keluar dari jendela cara bermain. Antarmuka jendela cara bermain ditunjukkan pada gambar 3.21.



Gambar 3.21 Tampilan antarmuka jendela cara bermain

3.3.1.2.4 Antarmuka Jendela About

Antarmuka jendela *about* menampilkan informasi tentang pembuatan game. Terdapat juga tombol *tutup* untuk keluar dari jendela *about* Antarmuka jendela about ditunjukkan pada gambar 3.22



Gambar 3.22 Tampilan antarmuka jendela *about*

BAB IV

ANALISIS KINERJA PERANGKAT LUNAK

4.1 Pengujian dan Analisis

Sebuah perangkat lunak belum bisa dikatakan bagus apabila belum dilakukan pengujian. Pengujian ini digunakan untuk mengetahui kekurangan pada perangkat lunak yang telah dibuat. Maka untuk mengetahui kekurangan pada *game* yang telah dibuat, sebanyak limabelas orang dengan usia antara 15 sampai 30 tahun diminta untuk memainkan *game* tersebut. Usia 15 sampai 30 tahun dipilih dengan asumsi bahwa kisaran umur tersebut tidak memiliki perbedaan yang signifikan dalam hal berpikir melawan komputer. Sehingga perbedaan penilaian untuk tingkat kesulitan hanya dibedakan dari sering atau tidaknya responden memainkan permainan halma yang dibagi menjadi tiga kategori : sering, kadang dan belum pernah sama sekali. Masing-masing responden bermain sebanyak lima kali. Setelah mereka memainkan *game* tersebut baru dibagikan kuisioner yang harus mereka isi. Kuisioner ini digunakan untuk mengetahui kekurangan dari *game* yang telah dibuat. Untuk melihat seperti apa kuisionernya dapat dilihat pada lampiran di halaman belakang.

4.2 Hasil Analisis

Dari kuisioner tersebut diberikan beberapa nilai untuk memudahkan penghitungan hasil analisisnya, yaitu :

Nilai 1 untuk jawaban sangat kurang

Nilai 2 untuk jawaban kurang

Nilai 3 untuk jawaban sedang

Nilai 4 untuk jawaban baik

Nilai 5 untuk jawaban sangat baik

Dari nilai tersebut dapat digunakan untuk menghitung nilai rata-rata dari jawaban responden. Rumus untuk menghitung nilai rata-ratanya adalah :

$$\text{Rata-rata} : \frac{\Sigma \text{ nilai jawaban (jumlah nilai jawaban)}}{\Sigma \text{ responden (jumlah responden)}}$$

Tabel 4.1 adalah hasil analisis

Jwbn responden / anyaan	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Rata2
npilan antarmuka?	5	4	4	3	3	4	3	3	3	3	5	3	3	4	3	3,53
gkat kesulitan?	4	3	4	4	3	4	3	3	3	4	5	4	3	3	4	3,67
nghibur?	5	4	5	4	4	3	4	4	4	4	4	3	3	4	4	3,93
mudahan ahami petunjuk?	4	4	4	5	4	3	3	5	4	3	5	4	4	4	5	4,06

4.2.1 Analisis Kesesuaian

Sesuai dengan teori dalam pembuatan *game* pada bab II, analisis yang dilakukan adalah :

1. *Game board*

   Game board merupakan bentuk tampilan. Menurut responden tampilan antarmuka *game* ini baik. Hal itu dapat dilihat dari tabel diatas pada pertanyaan no.1 tentang tampilan yang mempunyai nilai rata-rata yaitu: 3,53.

2. Tingkat kesulitan secara umum

   Menurut responden, tingkat kesulitan dalam *game* ini sedang, sesuai dengan nilai rata-rata yang diberikan responden pada pertanyaan nomer 2 yaitu 3,67

3. Nilai Hiburan

   Menurut responden *game* ini sudah cukup baik menghibur dengan nilai rata-rata yang diberikan 3,93.

4. Informasi untuk *player*

   Informasi pada *game* menurut responden sudah baik sesuai dengan nilai yang diberikan yaitu 4,06. Informasi pada *game* ini meliputi petunjuk permainan dan informasi pada saat permainan berjalan yaitu informasi giliran pemain dan informasi pemenang serta petunjuk bermain pada status bar di bawah papan permainan.

4.2.2 Analisis Tingkat Kesulitan Komputer

Untuk menganalisis tingkat kesulitan atau kecerdasan komputer diadakan pengujian khusus yang melibatkan tiga responden. Ketiga responden tersebut memiliki tingat pengalaman yang berbeda-beda terhadap permainan halma. Responden pertama menyukai permainan halma dan sangat sering memainkan permainan tersebut. Responden kedua mengetahui tentang permainan halma dan kadang-kadang memainkannya. Sedangkan responden ketiga belum pernah memainkan permainan halma sama sekali. Masing-masing responden memainkan *game* Halma sebanyak sepuluh kali dengan tingkat kesulitan komputer diset pada tingkat *sulit* dan kemudian dicatat hasil permainannya

1. Responden pertama

   Karena sudah sering memainkan permainan halma responden pertama tidak mengalami kesulitan dalam memahami peraturan dan cara bermain halma. Dari sepuluh babak permainan responden pertama dapat memenangkan tujuh babak dan mengalami kekalahan pada permainan ke 1, 4 dan 5. Pada ketiga babak tersebut responden pertama berada pada urutan pemenang ke dua.

2. Responden kedua

   Responden kedua memahami peraturan halma dan kadang-kadang memainkan permainan tersebut. Dari kesepuluh babak permainan responden kedua memenangkan 5 babak dan kalah pada babak 1, 3, 4, 6 dan 7. Pada babak tersebut responden kedua berada pada peringkat pemenang kedua.

3. Responden ketiga

   Responden ketiga belum pernah memainkan permainan halma sama sekali sehingga perlu mendapatkan sedikit penjelasan pada awal dimulainya permainan. Pada babak pertama responden mengalami kesulitan dan sering bertanya tentang cara menjalankan bidak halma. Responden memenangkan 4 dari 10 babak permainan. Pada babak 1, 2, 3 responden berada pada peringkat ke tiga dan pada babak 4, 6, 7 berada pada peringkat pemenang ke dua.

Dari hasil pengujian tersebut diketahui bahwa tingkat kecerdasan komputer tidak terlalu tinggi dan masih bisa dikalahkan oleh *pemain manusia* bahkan oleh yang belum pernah memainkan permainan halma sama sekali.

4.2.3 Analisis Perbandingan

Untuk mengetahui suatu *game* tersebut sudah baik apa belum selain digunakan kuisioner dapat pula digunakan perbandingan, yaitu dengan membandingkan *game* halma dengan *game* yang sudah ada. Dikarenakan tidak adanya atau sulitnya mencari *game* halma lain berupa software komputer, maka terpaksa perbandingan dilakukan terhadap *game* Hearts. *Game* ini dipilih karena mempunyai kesamaan dengan *game* Halma, yaitu memiliki lawan pemain komputer yang juga dilengkapi kecerdasan buatan.

Perbandingan *game* Halma dengan *game* Hearts ditampilkan pada tabel berikut:

Tabel 4.2 Hasil perbandingan

No.	Pembanding	*Game* Halma	*Game* Hearts
1.	Variasi level	Ada	Tidak Ada
2.	Variasi background	Tidak Ada	Tidak Ada
4.	Variasi Jumlah Player	Ada	Tidak Ada
5.	Keterangan waktu	Tidak Ada	Tidak Ada
6.	Efek Suara.	Tidak ada	Ada
7.	Informasi permainan	Ada	Ada

4.2.4 Kelebihan dan Kekurangan Program

Dari hasil perbandingan tersebut, ternyata ditemukan beberapa kelebihan dan kekurangan dari *game* Halma yang dibuat.

Kelebihan *game* Halma ini diantaranya adalah :

1. Tampilan antarmuka dan komposisi warnanya sudah cukup baik.

2. Mempunyai tingkat kesulitan *pemain komputer* yang cukup baik dengan tiga variasi tingkat kesulitan yang berbeda.
3. Dapat dimainkan oleh pemain manusia maupun komputer.

Kekurangan *game* Halma ini diantaranya adalah :

1. Pergerakan bidak belum di animasikan
2. Kecerdasan *pemain komputer* yang hanya menggunakan pencarian rute optimum mudah dikalahkan oleh pemain manusia.
3. Belum adanya efek suaranya.

BAB V

PENUTUP

5.1 Kesimpulan

Setelah dilakukan pengujian game tersebut, maka dapat disimpulkan sebagai berikut :

1. Dalam pembuatan *game*, yang harus dilakukan adalah pembuatan konsep *game* itu sendiri dan melakukan perancangan dengan menggunakan model yang mendukung pemrograman berorientasi obyek, sehingga dalam pembuatannya sudah tersedia kerangka kerjanya. Hal ini dapat membantu dalam menentukan batasan dan fokus pengembangannya.
2. Rata-rata responden berpendapat aplikasi *game* Halma yang dibuat cukup menarik pada tampilannya.
3. Sesuai dengan hasil analisis kuesioner yang diberikan pada responden, aplikasi *game* Halma yang dibuat cukup menghibur.
4. Rata-rata responden berpendapat tingkat kesulitan *pemain komputer* cukup baik, dengan tiga variasi tingkat kesulitan yang berbeda.
5. Menurut responden petunjuk cara bermain dan informasi yang desediakan pada *game* halma cukup mudah dimengerti.

5.2 Saran

Setelah melihat hasil yang dicapai dalam aplikasi *game* Halma ini, maka ada beberapa saran yang perlu diperhatikan, antara lain :

1. Mengembangkan aplikasi *game* Halma dengan menambahkan efek suara dan animasi untuk menambah keasyikan bermain *game*.
2. Mengembangkan aplikasi *game* Halma pada *mobile device*.
3. Menambahkan fitur jaringan pada pengembangan aplikasi *game* Halma..
4. Perlunya menyempurnakan kecerdasan buatan pada *pemain komputer* sehingga tidak hanya menggunakan pencarian rute optimum tetapi juga dapat memprediksi langkah lawan.

DAFTAR PUSTAKA

[DEW03] Dewi, Sri Kusuma. *Artificial Intelligence*, Yogyakarta : Graha Ilmu, 2003.

[HAR04] Hartanto, Antonius Aditya. *Pemrograman Mobile Java dengan MIDP 2.0*.Jogjakarta : ANDI , 2004.

[HAR04] Hariyanto, Bambang. *Rekayasa Sistem Berorientasi Objek*. Bandung : Informatika Bandung, 2004.

[HEN89] Hendra. *Membuat Program Permainan*. Jakarta : PT. Elex Media Komputindo,1989

[RIC03] Rickyanto, Isak. *Dasar Pemrograman Berorientasi Objek dengan Java2*. Jogjakarta : ANDI, 2003.

[SYA05] Syauqi, Taufikurrahman. *Mainan Pembunuh Waktu*. http://www.pikiran-rakyat.com/belia. Diakses pada tanggal 10 Februari 2007.

LAMPIRAN

HASIL PENGUJIAN

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : DT

Berapa usia anda sekarang? (..20... tahun)

Apakah anda suka bermain *game*? (Ya / ~~Kadang~~[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?					✓
2	Bagaimana tingkat kesulitan game tersebut?					✓
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?					✓

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

BELUM PERNAH

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : EMON

Berapa usia anda sekarang? (22. tahun)

Apakah anda suka bermain *game*? (Ya / ~~Kadang[2] / Tidak)~~*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah ~~/ Belum)~~

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			✓		
2	Bagaimana tingkat kesulitan game tersebut?			✓		
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?			✓		
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?				✓	

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

BLM PERNAH

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : S

Berapa usia anda sekarang? (..23.. tahun)

Apakah anda suka bermain *game*? (Ya / ~~Kadang~~[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (~~Pernah~~ / Belum)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			✓		
2	Bagaimana tingkat kesulitan game tersebut?				✓	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?			✓		

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : Fajar

Berapa usia anda sekarang? (18 tahun)

Apakah anda suka bermain *game*? (~~Ya~~ / Kadang[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			✓		
2	Bagaimana tingkat kesulitan game tersebut?			✓		
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?				✓	

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : ANTO

Berapa usia anda sekarang? (..24.. tahun)

Apakah anda suka bermain *game*? (Ya / ~~Kadang[2] / Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			✓		
2	Bagaimana tingkat kesulitan game tersebut?				✓	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?					✓

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

Belum pernah menemukan / memainkan game seperti ini, game classic yang bikin penasaran !

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : Bustomi

Berapa usia anda sekarang? (.22. tahun)

Apakah anda suka bermain *game*? (~~Ya~~ / Kadang[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			✓		
2	Bagaimana tingkat kesulitan game tersebut?			✓		
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?			✓		

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : TEGUH

Berapa usia anda sekarang? (24... tahun)

Apakah anda suka bermain *game*? (Ya / ~~Kadang~~[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?				✓	
2	Bagaimana tingkat kesulitan game tersebut?				✓	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?			✓		
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?			✓		

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

—

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : Husien. Widodo

Berapa usia anda sekarang? (..28.. tahun)

Apakah anda suka bermain *game*? (~~Ya~~ / Kadang[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			v		
2	Bagaimana tingkat kesulitan game tersebut?				v	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				v	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?					v

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : Rani

Berapa usia anda sekarang? (..21.. tahun)

Apakah anda suka bermain *game*? (~~Ya~~ / Kadang[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?				✓	
2	Bagaimana tingkat kesulitan game tersebut?				✓	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?					✓
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?				✓	

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

belum pernah

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : ENIRO

Berapa usia anda sekarang? (..26.. tahun)

Apakah anda suka bermain *game*? (~~Ya~~ / ~~Kadang~~[2] / Tidak)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

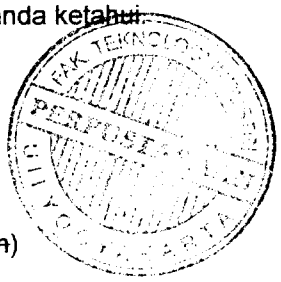
No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?				✓	
2	Bagaimana tingkat kesulitan game tersebut?			✓		
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?				✓	

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?
   BELOM PENAH MAIN HALMA DI KOMPUTER.

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.



Nama : Piz

Berapa usia anda sekarang? (...23... tahun)

Apakah anda suka bermain *game*? (~~Ya~~ / Kadang[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?					V
2	Bagaimana tingkat kesulitan game tersebut?				V	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?					V
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?				V	

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

Pelur pernah main game sama di komputer.

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : _____

Berapa usia anda sekarang? (..22.. tahun)

Apakah anda suka bermain *game*? (~~Ya~~ / Kadang[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			✓		
2	Bagaimana tingkat kesulitan game tersebut?				✓	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?					✓

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

   _____
   -
   _____
   _____
   _____

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : ANDRE

Berapa usia anda sekarang? (20 tahun)

Apakah anda suka bermain *game*? (Ya / ~~Kadang~~[2] / ~~Tidak~~)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / ~~Belum~~)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?			✓		
2	Bagaimana tingkat kesulitan game tersebut?				✓	
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?			✓		
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?				✓	

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

belum pernah main game halma di komputer

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

KUESIONER

Petunjuk Pengisian :
1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi).
2. Berilah tanda (v) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui.

Nama : Anah

Berapa usia anda sekarang? (..22.. tahun)

Apakah anda suka bermain *game*? (Ya / Kadang[2] / Tidak)*pilih salah satu

Apakah anda pernah bermain permainan Halma sebelumnya? (Pernah / Belum)

No	Pertanyaan	Nilai				
		1	2	3	4	5
1	Bagaimana tampilan antarmuka *game* Halma yang Anda uji coba?				✓	
2	Bagaimana tingkat kesulitan game tersebut?			✓		
3	Apakah Anda merasa terhibur ketika memainkan *game* Halma?				✓	
4	Bagaimana tingkat kemudahan dalam memahami aturan dan memainkan *game* Halma?				✓	

5. Jika Anda pernah memainkan *game* halma selain *game* ini sebelumnya, menurut Anda apakah kelebihan *game* Halma ini dibandingkan dengan yang pernah Anda mainkan?

gak pernah main game halma di komputer

Keterangan Nilai : 1. Sangat Kurang
2. Kurang
3. Sedang
4. Baik
5. Sangat Baik

PENGUJIAN TINGKAT KESULITAN KOMPUTER

- Responden 1

  Nama : Dewi

  Umur : 25

  Jenis klmn : P

  Tingkat keseringan bermain Halma : (Sering / ~~Kadang2 / Belum Pernah~~)*

  Ket : bermain halma menjadi salah satu hobby

Tabel Pengamatan 1

Babak	Menang? (V)	Peringkat
1		2
2	✓	
3	✓	
4		2
5		2
6	✓	
7	✓	
8	✓	
9	✓	
10	✓	

- Responden 2

  Nama : Anton

  Umur : 28

  Jenis klmn : L

  Tingkat keseringan bermain Halma : (~~Sering~~ / Kadang2 / ~~Belum Pernah~~)*

  Ket : untuk mengisi waktu kadang-kadang main halma.

Tabel Pengamatan 2

Babak	Menang? (V)	Peringkat
1		2
2	✓	
3		2
4		2
5	✓	
6		2
7		2
8	✓	
9	✓	
10	✓	

- Responden 3

  Nama : Totok

  Umur : 26

  Jenis klmn : L

  Tingkat keseringan bermain Halma : (~~Sering / Kadang2~~ / Belum Pernah)*

  Ket : Belum pernah bermain halma sama sekali

Tabel Pengamatan 3

Babak	Menang? (V)	Peringkat
1		3
2		3
3		3
4		2
5	✓	
6		2
7		2
8	✓	
9	✓	
10	✓	