 ISSN : 1858 – 3938

PERMAINAN KARTU REMI 41 DENGAN MENGGUNAKAN ALGORITMA *LINEAR CONGRUENTIAL GENERATOR* (LCG) SEBAGAI PEMBANGKIT ANGKA RANDOM

Nixon Erzed[1)] Anggun Rospita[2)]

[1)] Nixon Erzed
Sekolah Tinggi Teknologi Informasi NIIT I-Tech
Jl. Asem 2 No.22, Cipete – Jakarta Selatan
http://www.i-tech.ac.id
nixon_rz@yahoo.com

[2)] Anggun Rospita
Sekolah Tinggi Teknologi Informasi NIIT I-Tech
Jl. Asem 2 No.22, Cipete – Jakarta Selatan
http://www.i-tech.ac.id
nggunross@ymail.com

ABSTRAKSI

Selain digunakan sebagai media untuk *refreshing*, game atau permainan sering kali digunakan untuk media pembelajaran dan pelatihan peningkatan *kognitif*. Salah satu game yang dikategorikan memiliki tingkat kesulitan yang cukup tinggi adalah permainan kartu remi 41 karena pemain harus cerdik dan pandai dalam menghitung nilai kartu serta peluang yang muncul saat permainan berlangsung. Di dalam mengimplentasikan permainan kartu remi 41 dibutuhkan algoritma yang dapat menunjang terbentuknya aplikasi sesuai dengan kebutuhan permainan. Kebutuhan utama dalam permainan kartu adalah proses pengacakan kartu di awal permainan dan jalannya permainan itu sendiri. Pemanfaatan algoritma *Linear Congruential Generator* (LCG) dirasa sesuai dengan kebutuhan proses pembangkitan angka random untuk pengacakan kartu.Dalam pengimplementasian game kartu remi 41 secara *offline* dan *single player*, peranan *Artificial Intelligent* sangatlah penting dalam jalannya permainan. Kecerdasan buatan yang digunakan harus bisa mengimbangi kualitas permainan dari *player* (manusia). Permainan kartu remi 41 yang dikembangkan terbukti dapat diimplementasikan dengan baik menggunakan Algoritma Linear Congruential Generator sebagai pembangkit kartu acak yang tidak berulang.

PENDAHULUAN

Game atau permainan merupakan kegiatan yang disukai oleh masyarakat di penjuru dunia maupun masyarakat Indonesia. Salah satu game yang dikategorikan memiliki tingkat kesulitan yang cukup tinggi adalah permainan kartu remi. Permainan kartu remi dikategorikan sebagai permainan dengan tingkat kesulitan tinggi karena pemain harus cerdik dan pandai dalam menghitung nilai kartu serta peluang yang muncul saat permainan berlangsung. Salah satu permainan remi yang populer di Indonesia adalah permainan kartu remi 41. Dilihat dari segi permainan, permainan kartu remi 41 bersifat kompetitif sehingga membutuhkan strategi untuk mengalahkan lawan. Ini akan menstimulasi aspek kognitif sehingga sedikit banyak permainan tersebut dapat memperkaya kemampuan berpikir. Permainan kartu remi dapat diimplementasikan menjadi aplikasi game baik bersifat online maupun bersifat offline. Di dalam mengimplentasikan permainan kartu remi 41 dibutuhkan algoritma yang dapat menunjang terbentuknya aplikasi sesuai dengan kebutuhan permainan. Kebutuhan utama dalam permainan kartu ada proses pengacakan kartu di awal permainan dan jalannya permainan itu sendiri. Dalam kegiatan pengacakan kartu terdapat proses pembangkitan angka random yang hasilkan akan di sinkronisasi dengan nama kartu, sehingga tercipta pengacakan kartu yang optimal. Pengacakan kartu dikatakan berhasil apabila tidak ada kartu yang muncul lebih dari satu kali dan jumlah kartu yang teracak sesuai dengan jumlah kartu yang digunakan. Sebagaimana diketahui kartu remi merupakan kartu yang terdiri dari 52 set kartu dengan kombinasi 4 jenis kartu yaitu spade (♠), heart (♥), diamond (♦) dan Club (♣). Dari empat

jenis itu terdapat 13 buah nilai kartu mulai As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen dan King. Pemanfaatan algoritma Linear Congruential Generator (LCG) dirasa sesuai dengan kebutuhan proses pembangkitan angka random untuk pengacakan kartu. Secara teoritis, algoritma LCG mampu menghasilkan bilangan acak yang baik, namun sangat sensitif terhadap pemilihan nilai-nilai a, b, dan m pada relasi rekurensnya. Pemilihan nilai-nilai yang tidak sesuai dapat mempengaruhi implementasi pada LCG. Selain proses pengacakan kartu, proses penting lainnya yang memegang peranan penting dalam pengaplikasian game kartu remi 41 adalah proses jalannya permainan seorang player melawan player yang lain. Dalam pengimplementasian game kartu remi 41 secara offline dan single player, peranan Artificial Intelligent sangatlah penting dalam jalannya permainan. Kecerdasan buatan yang digunakan harus bisa mengimbangi kualitas permainan dari player (manusia)

TEORI DASAR

Rekayasa Perangkat Lunak

Istilah Rekayasa Perangkat Lunak (RPL) secara umum disepakati sebagai terjemahan dari istilah *Software Engineering*. Istilah *Software Engineering* mulai dipopulerkan tahun 1968 pada *Software Engineering Conference* yang diselenggarakan oleh NATO. Sebagian orang mengartikan RPL hanya sebatas pada bagaimana membuat program komputer. Padahal ada perbedaan yang mendasar antara perangkat lunak *(software)* dan program komputer. Perangkat lunak adalah seluruh perintah yang digunakan untuk memproses informasi. Perangkat lunak dapat berupa program atau prosedur. Program adalah kumpulan perintah yang dimengerti oleh komputer sedangkan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi. Pengertian RPL sendiri adalah sebagai berikut : *"Suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan."* Jelaslah bahwa RPL tidak hanya berhubungan dengan cara pembuatan program komputer. Pernyataan "semua aspek produksi" pada pengertian di atas, mempunyai arti semua hal yang berhubungan dengan proses produksi seperti manajemen proyek, penentuan personil, anggaran biaya, metode, jadwal, kualitas sampai dengan pelatihan pengguna merupakan bagian dari RPL. **Model Air Terjun *(Waterfall)*** Merupakan paradigma rekayasa perangkat lunak yang paling tua dan paling banyak dipakai. Model ini mengusulkan sebuah pendekatan perkembangan perangkat lunak yang sistematik dan sekunsial yang dimulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain , kode, pengujian, dan pemeliharaan. Seiring dengan perkembangan sistem informasi saat ini model ini senantiasa berkembang menyesuaikan perkembangan yang ada. Model *Waterfall* mengikuti aktivitas-aktivitas yaitu:

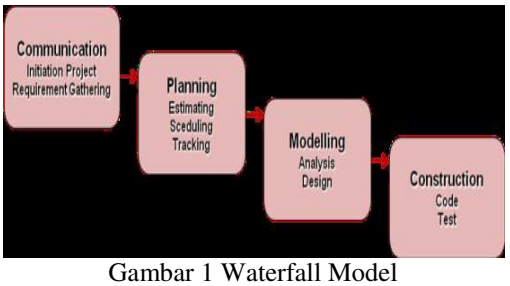
a) Komunikasi
Inisiasi proyek dan *requirements gathering* adalah aktifitas yang dikerjakan di tahap paling awal. Karena pada tahap ini akan dibentuk suatu relasi yang kuat membahas tentang apa saja masalah yang dapat di bantu dan diselesaikan oleh sistem yang akan dibuat.

b) Perencanaan
Setelah terjalin komunikasi yang baik dan ditemukannya masalah-masalah yang dapat dibantu dan diselesaikan, tahapan masuk ke perencannaan sehingga estimasi waktu, penjadwalan proyek dan *tracking* proyek dapat di buat.

c) Permodelan
Analisis Kebutuhan Perangkat Lunak : Proses menganalisis dan pengumpulan kebutuhan sistem yang sesuai dengan domain informasi tingkah laku, unjuk kerja, dan antar muka *(interface)* yang diperlukan. Kebutuhan-kebutuhan tersebut didokumentasikan dan di lihat lagi dengan pelanggan.
Desain Proses : Desain akan menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi *interface*, dan detail (algoritma) prosedural.

d) Pembangunan Pemprograman
Pengkodeaan (*Coding*) : Pengkodean merupakan proses menerjemahkan desain ke dalam suatu bahasa yang bisa dimengerti oleh komputer. Pengujian : Proses pengujian dilakukan pada logika internal untuk memastikan semua pernyataan sudah diuji. Pengujian eksternal fungsional untuk menemukan kesalahan-kesalahan dan memastikan bahwa input akan memberikan hasil yang aktual sesuai yang dibutuhkan

 ISSN : 1858 – 3938



Gambar 1 Waterfall Model

Pemrograman Berbasis Objek Dengan Java

Bahasa pemrograman Java saat ini popularitasnya meningkat untuk be berapa tahun belakangan ini.Banyak *game* dan aplikasi yang digunakan pada perangkat *mobile* seperti telepon seluler dan PDA di buat dengan bahasa ini. Bahasa ini dikenal karena portabilitas dan dukungan pada konsep pemrograman berorientasi objek. Terdapat dua standart kompetensi, yaitu program dalam bahasa pemrograman berorientasi objek dan program aplikasi menggunakan Java. Hal ini karena kedekatan konsep antara Java dan pemrograman berorientasi obyek. Standart kompetensi membuat program dalam bahasa pemrograman berorientasi obyek terdiri dari emapt kompetensi dasar yaitu tipe data dan control program, pembuatan kelas, penggunakaan *inheritance*, *polymorphism* dan *overloading* dan penggunaan interface dan paket. Sedangkat standart kompetensi membuat program aplikasi menggunakan Java terdiri dari lima kompetensi dasar, yaitu menjelaskan tentang file I/O, tipe data dan variabel, menerapkan operator, menjelaskan *exception handling*, menerapkan *multithreading* dan menjelaskan *network programming.*

Pembangkit Bilangan Acak Semu

Bilangan acak *(random)* banyak digunakan di dalam kriptografi, misalnya untuk pembangkitan elemen-elemen kunci (contohnya pada algoritma OTP), pembangkitan *initialization vector (IV),* pembangkitan parameter kunci di dalam sistem kriptografi kunci publik, dan sebagainya. Yang dimaksud dengan acak disini adalah bilangan yang tidak mudah diprediksi oleh pihak lawan. Sayangnya sangat sulit memperoleh bilangan acak dalam paktek kriptografi. Tidak ada prosedur komputasi yang benar-benar menghasilkan deret bilangan acak yang sempurna. Bilangan acak yang dihasilkan dengan rumus-rumus matematika adalah bilangan acak semu (*pseudo*) karena bilangan acak yang dibangkitkan dapat berulang kembali secara periodik. Pembangkit deret bilangan acak semacam itu disebut *pseudo-random number generator (PRNG)* ***Linear Congruential Generator (LCG)*** Pembangkit bilangan acak kongruen-lanjar (*linear congruential generator* atau *LCG*) adalah salah satu pembangkit bilangan acak tertua dan sangat terkenal. LCG didefinisikan dalam relasi rekurens : $\mathbf{x_n = (ax_{n-1} + b) \bmod m}$ Yang dalam hal ini, Xn = bilangan acak ke-n dari deretnya Xn-1 = bilangan acak sebelumnya a = faktor pengali b = increment m = modulus (a, b, dan m semuanya konstanta) LCG mempunyai periode tidak lebih besar dari m, dan pada kebanyakan kasus periodenya kurang dari itu. LCG mempunyai periode penuh (m-1) jika memenuhi syarat berikut :

1 b relatif prima terhadap m
2 a-1 dapat dibagi dengan semua faktor prima dari m
3 a-1 adalah kelipatan dari 4 jika m adalah kelipatan dari 4
4 m > maks (a, b, x0)
5 a > 0, b > 0

Meskipun LCG secara teoritis mampu menghasilkan bilangan acak yang lumayan, namun ia sangat sensitif terhadap pemilihan nilai – nilai a, b dan m. Pemilihan nilai – nilai yang buruk dapat mengarah pada implementasi LCG yang tidak bagus. Keunggulan LCG terletak pada kecepatannya dan hanya membutuhkan sedikit operasi bit. Sayangnya, LCG tidak dapat digunakan untuk kriptografi karena bilangan acaknya dapat diprediksi urutan kemunculannya. Oleh karena itu LCG tidak aman digunakan untuk kriptografi. Namun demikian, LCG tetap berguna untuk aplikasi non-kriptografi seperti simulasi, sebab LCG mangkus dan memperlihatkan sifat statistik yang bagus dan sangat tepat untuk uji-uji empirik.

Diagram Use Case

Use Case Diagram menggambarkan fungsionalitas dari sistem dan *scenario* yang menjelaskan mengenai interaksi antara user dengan sistem. Sebuah *diagram use case* melukiskan :

1 Aktor
Aktor bukan merupakan bagian dari sistem, aktor merepresentasikan orang atau sesuatu yang berinteraksi dengan sistem.
2 *Use Case*
Use case memodelkan sebuah komunikasi antara aktor dan sistem. Use case menggambarkan fungsionalitas yang disediakan oleh sistem, yaitu kemampuan apa saja yang disediakan kepada aktor oleh sistem.
3 *Generalisasi*
Generalisasi, digambarkan dengan garis langsung dengan sebuah segitiga kosong*(hollow triangle).*Kata generalisasi pada UML berarti *"inheritance".* Ketika kita menemukan sebuah relasi generalisasi antar dua actor atau dua *use*

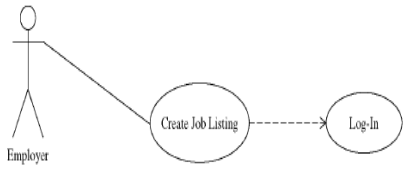
 ISSN : 1858 – 3938

case, maka itu mengindikasikan *child actor* atau *use case* adalah sebuah *instance* dari base *actor* atau *use case*

Gambar 2 Generalisasi

4 *Dependensi*

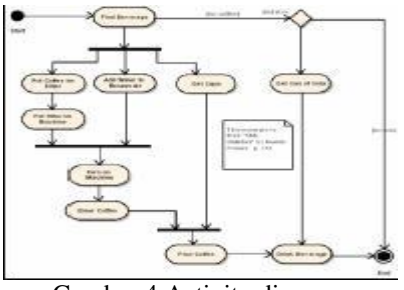
Dependensi, digambarkan dengan garis putus putus-putus dengan panah penunjuk *(directional arrow).* Arah panah menandakan use case bergantung pada(depended on)



Gambar 3 Dependensi

Activity Diagram

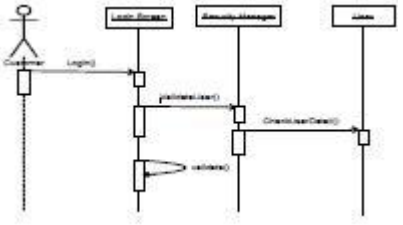
Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang,bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. Contoh *activity diagram* tanpa *swimlane*:



Gambar 4 Activity diagram

Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek didalam dan disekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metode dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.



Gambar 5 Sequence diagram

Artificial Intellegent

Kecerdasan Buatan atau *Artificial Intelligence* atau AI didefinisikan sebagai kecerdasan entitas ilmiah. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (games), logika *fuzzy*, jaringan syaraf tiruan dan

 ISSN : 1858 – 3938

robotika. Banyak hal yang kelihatannya sulit untuk kecerdasan manusia, tetapi untuk Informatika relatif tidak bermasalah. Seperti contoh: mentransformasikan persamaan, menyelesaikan persamaan integral, membuat permainan catur atau Backgammon. Di sisi lain, hal yang bagi manusia kelihatannya menuntut sedikit kecerdasan, sampai sekarang masih sulit untuk direalisasikan dalam Informatika. Seperti contoh: Pengenalan Obyek/Muka, bermain sepak bola. Walaupun AI memiliki konotasi fiksi ilmiah yang kuat, AI membentuk cabang yang sangat penting pada ilmu komputer, berhubungan dengan perilaku, pembelajaran dan adaptasi yang cerdas dalam sebuah mesin. Penelitian dalam AI menyangkut pembuatan mesin untuk mengotomatisasikan tugas-tugas yang membutuhkan perilaku cerdas. Termasuk contohnya adalah pengendalian, perencanaan dan penjadwalan, kemampuan untuk menjawab diagnosa dan pertanyaan pelanggan, serta pengenalan tulisan tangan, suara dan wajah. Hal-hal seperti itu telah menjadi disiplin ilmu tersendiri, yang memusatkan perhatian pada penyediaan solusi masalah kehidupan yang nyata. Sistem AI sekarang ini sering digunakan dalam bidang ekonomi, obat-obatan, teknik dan militer, seperti yang telah dibangun dalam beberapa aplikasi perangkat lunak komputer rumah dan video game. 'Kecerdasan buatan' ini bukan hanya ingin mengerti apa itu sistem kecerdasan, tapi juga mengkonstruksinya. ***Game*** Game adalah sebuah aktivitas rekreasi dengan tujuan bersenang senang, mengisi waktu luang, atau berolahraga ringan dan kadang juga dipergunakan sebagai alat pembelajaran. Permainan biasanya dilakukan sendiri atau bersama-sama. Saat ini game bisa dikatakan sebagai permainan *universal* , seseorang dapat memainkan dimana saja dan kapan saja mungkin inilah yang membuat banyak kalangan yang menggeluti dunia game yang semakin lama dari waktu ke waktu teknologi game itu sendiri semain canggih dan mendekati dengan real. Berikut ini akan dijelaskan bagaimana sejarah dari game yang pertama kali dibuat:

1. Game yang pertama kali dibuat adalah tictactoe pada tahun 1952 yang dikembangkan oleh A.S.Douglas yang dimainkan pada *vaccum tube computer.*
2. .Selanjutnya pada tahun 1958 dengan nama tenis for two yang di kembangan oleh Willy Higginbothman yang dimainkan pada *osciloscope* yang dihubungkan dengan *analog donner computer*.
3. Pada tahun 1960an (1960-1970) di tahun 1961-1962 munculah game Space War yang dikembangkan oleh MIT dengan menggunkan *vector graphic* yang dimainkan pada komputer PDP-1, kemudian sega mengeluarkan *game arcade* pertama yaitu *electronic shooting game*.
4. Lalu pada tahun 1971, Nolan Bushel (Nutting) *Develops Computer Space*. Ini adalah jenis game *arcade* komersial pertamakali yaitu dengan basic dasar pengembangannya pada SpaceWar. Graphics dengan vector graphics tapi sayangnya game ini gagal dipasaran.
5. Pada tahun 1972 Bushnell memulainya dengan Atari pertama kali dikeluarkan harganya $100/console .
6. Pada tahun 1972-1976, munculah game adventure dengan judul The Colossal Cave, game ini cara mainnya berdasarkan teks, dan saat itu dijalankan dengan DEC *mainframe*.

Deskripsi Permainan

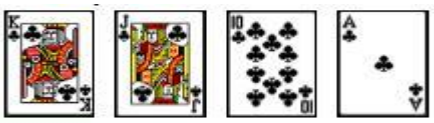
Permainan ini menerapkan sistem pengacakan / *random* kartu. Pemain dituntut untuk mengerti konsep dasar permainan kartu remi 41 terlebih dahulu. Dalam pemrosesannya setiap pemain akan mendapatkan sejumlah kartu pada saat permainan dimulai. Kemudian selama permainan berlangsung, pemain mengkombinasikan kartu yang mereka miliki sehingga memperoleh nilai 41 dengan jenis dan warna kartu yang sama. Kombinasi tersebut diperoleh dari 4 kartu yang dimiliki oleh pemain. Masing – masing pemain memiliki kesempatan untuk mengambil kartu dari tumpukan sesuai dengan giliran. Bila pemain telah mengambil satu kartu dari tumpukan, maka ia wajib membuang salah satu kartunya yang dianggap tidak penting atau memiliki poin terkecil. Sehingga dari awal hingga akhir permainan, masing-masing pemain hanya memiliki 4 kartu terpegang. Pemain yang terlebih dahulu mendapatkan nilai 41 pada kartunya, maka ia berhak menjadi pemenang. Apabila hingga akhir permainan (kondisi dimana kartu pada tumpukan telah habis) belum ada pemain yang mencapi poin 41, maka pemenang ditentukan dari pemilik poin tertinggi dari kartu yang mereka miliki Permainan diawali dengan pembagian kartu kepada 4 pemain. Masing-masing pemain mendapatkan 4 kartu acak. Setelah menerima kartu, pemain melakukan pengecekan nilai kartu yang mereka miliki. Nilai kartu dianggap “sah” atau “jadi” apabila jenisnya sama (warna dan bentuk). Nilai masing-masing kartu adalah sebagai berikut :

 ISSN : 1858 – 3938

Tabel 1 Tabel Nilai Poin Kartu

Jenis Kartu	Nilai Poin	Jenis Kartu	Nilai Poin
As	11	7	7
King	10	6	6
Queen	10	5	5
Jack	10	4	4
10	10	3	3
9	9	2	2
8	8		

Poin yang dikumpulkan bernilai 41 untuk kartu yang sejenis, contohnya :

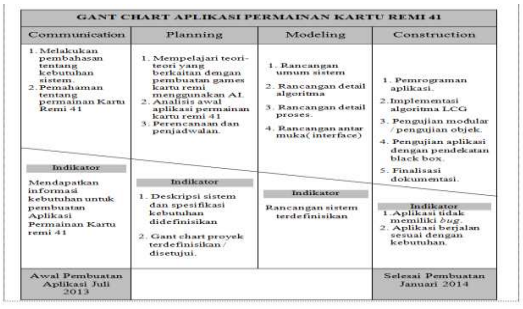


Gambar 6 Susunan kartu yang menghasilkan poin 41

Permainan berhenti apabila terjadi salah satu diantara dua kondisi berikut :

1. **Salah satu pemain telah berhasil mendapatkan poin 41**
   Pemain yang terlebih dahulu mendapatkan poin 41 adalah pemenangnya

2. **Kartu ditumpukan telah habis**
   Apabila kartu di tumpukan telah habis, maka otomatis permainan akan berakhir. Penentuan pemenang dilihat dari jumlah poin yang dikumpulkan masing-masing pemain. Nilai kartu yang sejenis dijumlahkan dan dikurangi nilai kartu yang berbeda jenis. Pemain dengan nilai tertinggi menjadi pemenangnya.



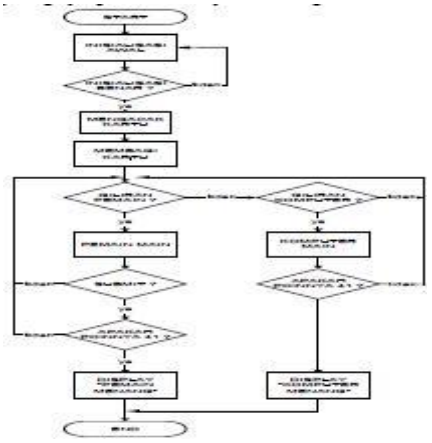
GANT CHART APLIKASI PERMAINAN KARTU REMI 41

Communication	Planning	Modeling	Construction
1. Melakukan pembahasan tentang kebutuhan sistem. 2. Pemahaman tentang permainan Kartu Remi 41	1. Mempelajari teori-teori yang berkaitan dengan pembuatan games kartu remi menggunakan AI. 2. Analisis awal aplikasi permainan kartu remi 41. 3. Perencanaan dan penjadwalan.	1. Rancangan umum sistem 2. Rancangan detail algoritma 3. Rancangan detail proses. 4. Rancangan antar muka(interface)	1. Pemrograman aplikasi. 2.Implementasi algoritma LCG 3. Pengujian modular / pengujian objek. 4. Pengujian aplikasi dengan pendekatan black box. 5. Finalisasi dokumentasi.
Indikator Mendapatkan informasi kebutuhan untuk pembuatan Aplikasi Permainan Kartu remi 41	Indikator 1. Deskripsi sistem dan spesifikasi kebutuhan didefinisikan 2. Gant chart proyek terdefinisikan / disetujui.	Indikator Rancangan sistem terdefinisikan	Indikator 1. Aplikasi tidak memiliki *bug*. 2. Aplikasi berjalan sesuai dengan kebutuhan.
Awal Pembuatan Aplikasi Juli 2013			Selesai Pembuatan Januari 2014

Tabel 2 Gantt chart aplikasi

Flow Chart Umum

Flowchart umum merupakan gambaran umum proses berlangsungnya permainan, yaitu sebagai berikut :



Gambar 7 *Flowchart* Umum Permainan 41

Dalam proses pengacakan kartu, akan diterapkan penghitungan menggunakan Algoritma LCG (*Linear Congruential Generator).*
Implementasi algoritma LCG ini dalam proses pengacakan kartu dalam permainan kartu remi 41 ini adalah sebagai berikut :

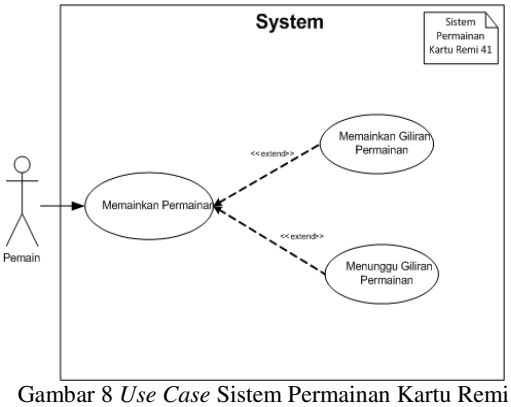
$$x_n = (a x_{n-1} + b) \bmod m$$

$$x_n = (13\, x_{n-1} + 7) \bmod 52$$

Dimana : a = 13, faktor pengali didapat dari banyaknya jenis nilai kartu remi (2,3,4,...,As) b = 7 , nilai yang relatif prima terhadap m karena faktor prima dari 7 (1 dan 7) merupakan faktor prima dari 52 (faktor prima 52 : 1,3,7) m = 52, modulus didapat dari banyaknya jumlah kartu

Use case Diagram

Usecase diagram aplikasi permainan kartu ini memiliki satu aktor yaitu : pemain, satu fungsionalitas utama yaitu : memainkan permainan, dengan dua fungsionalitas *extend.* Secara lebih detil adalah sebagai berikut :



Gambar 8 *Use Case* Sistem Permainan Kartu Remi 41

 ISSN : 1858 – 3938

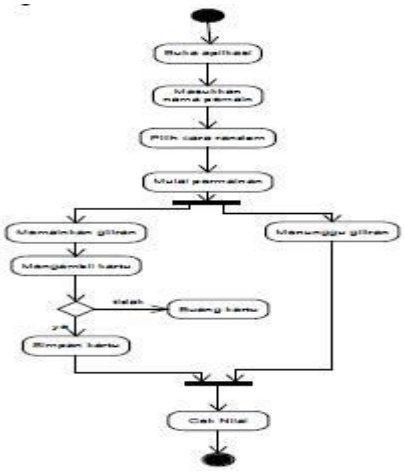
Skenario *Use case*

Berikut ini penjelasan skenario *Use case* Sistem Permainan Kartu Remi 41 tersebut.

1. **Nama *Use case*** ***:*** Memainkan Permainan
2. **Actor** **:** Pemain
3. **Precondition** **:** Menampilkan Aplikasi Permainan
4. **Postcondition** **:** Pemain dapat memulai permainan kartu remi 41 setelah mengisi data yang di butuhkan berupa nama pemain
5. **Description** **:** Permainan dimainkan setelah proses inisialisasi awal telah dilakukan. Mencakup nama pemain dan aturan permainan (pemilihan metode pengacakan). Permainan dimulai ketika sistem telah selesai mengacak dan membagi kartu pada masing-masing pemain. Pemain dapat segera memainkan dan menunggu giliran untuk bermain.

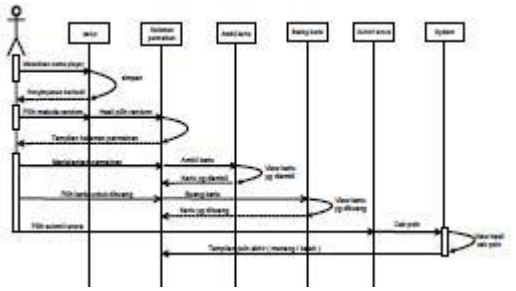
***Activity* Diagram**

Activity diagram merupakan diagram yang merepresentasikan langkah demi langkah berjalannya suatu sistem**.** *Activity* diagram merupakan state diagram khusus, di mana sebagian besar state adalah *action* dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya. Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Dibawah ini akan digambarkan action user pada sistem yang akan dibuat.



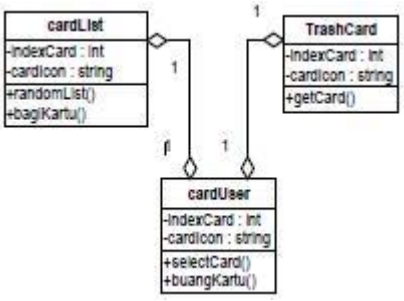
Gambar 9 *Activity* Diagram Proses Permainan

Berikut ini merupakan *sequence* diagram untuk permainan Pemain :



Gambar 10 *Sequence* Diagram Pemain

Dalam sequence diagram ini dijelaskan step awal yang dilakukan pemain adalah proses setup berupa input nama pemain. Selanjutnya pemain akan memilih metode pengacakan kartu (randomize), halaman permainan akan menampilkan halaman dengan tumpukan kartu sesuai dengan pilihan hasil pengacakan kartu dari pemain. Saat proses permainan berlangsung, pemain mengambil kartu dan membuang kartu, tampilan kartu yang dibuang dan diambil akan diberikan oleh sistem didalam papan permainan. Saat pemain memilih cek nilai (submit score) maka sistem akan menilai poin dari pemain tersebut dan menampilkannya dalam halaman permainan. Dalam pembangunan aplikasi permainan kartu remi 41 ini akan dibentuk beberapa *class* dengan atribut dan method maisng-masing, berikut ini adalah penjelasannya dalam bentuk *class diagram*.



Gambar 11 Class diagram Dalam class diagram

diatas dijelaskan 3 class yang ada, yaitu :

1. *CardList* - *CardList* adalah kelas untuk kartu tumpukan yang digunakan pemain yang didapat dari hasil pengacakan kartu di awal permainan. Dengan atribut *indexCard* (tipe data *integer*) dan *cardIcon* (tipe data *string*). *Method* yang dijalankan dalam *class CardList* adalah *randomList*( ) & bagiKartu().
2. *TrashCard* - *TrashCard* adalah kelas untuk kartu tumpukan yang telah dibuang oleh pemain. Dengan atribut *indexCard* (tipe data *integer*) dan *cardIcon* (tipe data *string*). *Method* yang dijalankan dalam *class TrashCard* adalah *getCard*( ).

 ISSN : 1858 – 3938

3. *CardUser* - *CardUser* adalah kelas untuk yang dipegang oleh pemain selama permainan berlangsung. Dengan atribut *indexCard* (tipe data *integer*) dan *cardIcon* (tipe data *string*). *Method* yang dijalankan dalam class *CardUser* adalah *selectCard*( ) dan buangKartu( ).

Lingkungan Implementasi

Dalam pembuatan permainan kartu remi 41 ini, akan dijelaskan beberapa hal yang berhubungan dengan hardware, software, serta komponen – komponen lain pada saat pembuatan game ini. Perangkat keras yang digunakan untuk menjalankan aplikasi adalah sebagai berikut :

1. Netbook ASUS dengan spesifikasi hardware sebagai berikut :
   a. Processor AMD E-450 APU with Radeon™ HD Graphics 1.6Ghz, RAM 2 GB, Harddrive 320 GB Hitachi HTS543 SATA Disk Device, Graphic Card AMD Radeon HD 6320 Graphics, Sistem Operasi Windows-7 Professional
2. Software yang digunakan adalah sebagai berikut :
   a. NetBeans IDE 7.4 - Digunakan sebagai script editor yang digunakan untuk proses scripting untuk game.
   b. Adobe Photoshop - Aplikasi ini digunakan untuk merancang tampilan antarmuka serta gambar – gambar yang ada pada aplikasi game.

Implementasi Sistem

Dalam aplikasi permainan kartu remi 41 ini terdapat 7 form antara lain :

1) Form Awal Permainan
2) Form Input Nama Pemain
3) Form Pemilihan Metode Pengacakan Kartu
4) Form Peraturan umum
5) Form Permainan Utama
6) Form Tampilan Nilai
7) Form Akhir Permainan

Form Awal permainan

Halaman ini muncul pada pertama kali saat pemain meload aplikasi game. Aktivitas loading ini merupakan proses eksekusi dari aplikasi game. Dalam form ini, pemain akan mengklik tombol "Main" untuk masuk ke dalam permainan.

Gambar 12 Tampilan halaman muka

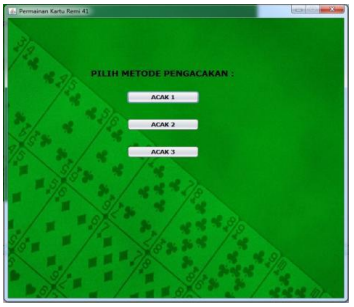
Form *Input* Nama Pemain

Form selanjutnya yang ditampilkan adalah form *input*, disini pemain harus memasukkan nama pemain dan mengklik tombol "Masuk". Tampilan *Form Input* Nama Pemain adalah sebagai berikut :

Gambar 13 Tampilan halaman input nama pemain

Form Pemilihan Metode Pengacakan Kartu

Form ini merupakan navigasi bagi pemain, didalam form ini terdapat 3 pilihan metode pengacakan kartu yang bisa digunakan. Pemain akan memilih salah satu metode pengacakan kartu yang akan digunakan dalam permainan. Tampilan form pemilihan metode pengacakan kartu adalah sebagai berikut :



Gambar 14 Halaman pemilihan metode pengacakan kartu

Form Peraturan Umum

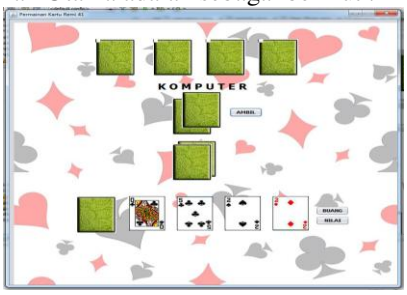
Form ini menampilkan peraturan umum dalam permainan 41. Dalam form ini dijelaskan tentang peraturan perolehan poin tertinggi dan maksimal waktu giliran pemain. Tampilan form peraturan umum adalah sebagai berikut :

Gambar 15 Halaman peraturan Umum

 ISSN : 1858 – 3938

Form Permainan Utama

Form ini merupakan form dimulainya permainan. Dalam form ini akan ditampilkan nama pemain serta kartu yang dipegang pemain. Sistem akan menampilkan notifikasi giliran bermain. Saat gilirannya bermain, pemain dapat mengklik tombol "Ambil" yang ada di samping tumpukan kartu untuk mengambil kartu teratas dari tumpukan. Apabila pemain ingin membuang salah satu kartu, pemain cukup klik "Buang". Apabila pemain merasa poin kartunya sudah mencukupi (sudah 41 poin) maka pemain dapat mengklik tombol "Nilai". Sistem akan memeriksa poin pemain, apabila poin belum tercukupi sistem akan menampilkan pemberitahuan kepada pemain. Tampilan Form Permainan Utama adalah sebagai berikut :



Gambar 16 Tampilan Form Permainan

Form Tampilan Nilai

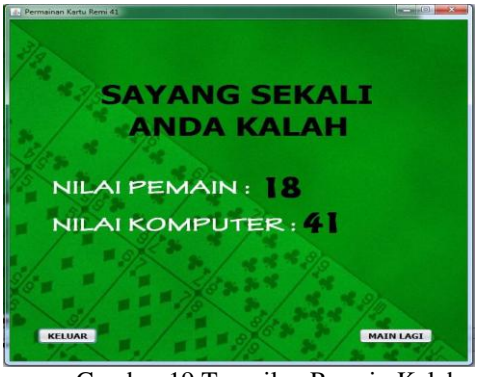
Dalam form ini akan ditampilkan nilai poin kartu dari pemain apabila pemain melakukan pengecekan nilai dengan menekan tombol "Nilai". Sistem akan melakukan pengecekan nilai dari kartu yang dipegang oleh pemain. Tampilan form pengecekan nilai adalah sebagai berikut :

Gambar 17 Tampilan Nilai

Form Akhir Permainan

Form akhir permainan ini akan menampilkan pemberitahuan hasil akhir permainan apakah pemain kalah atau menang. Form akhir permainan ini akan ditampilkan apabila permainan berakhir. Permainan berakhir karena dua kondisi yaitu salah satu pemain telah mendapatkan poin 41 atau kartu di tumpukan telah habis. Apabila salah satu dari kondisi itu terpenuhi, sistem akan otomatis melakukan pengecekan nilai dan menampilkannya dalam form akhir permainan. Tampilan form akhir permainan adalah sebagai berikut :

Gambar 18 Tampilan Pemain Menang



Gambar 19 Tampilan Pemain Kalah

Pengujian

Pengujian aplikasi ini menggunakan 2 metode pengujian yaitu *black box* dan *white box*. Pengujian *black box* yaitu pengujian yang berfokus pada persyaratan fungsional perangkat lunak. Sedangkan pengujian dengan metode *white box* adalah pengujian yang dilakukan berdasarkan fungsi-fungsi yang ada di dalam setiap form

Rencana Pengujian

Rencana pengujian selengkapnya dapat dilihat pada tabel berikut :

 ISSN : 1858 – 3938

Tabel 3 Rencana Pengujian

Pengujian	Jenis Pengujian
Input nama	Black Box
Pre Loader Loading	Black Box
Pengacakan Kartu	Black Box
Pembagian Kartu	Black Box
Fungsi tombol "Ambil"	Black Box
Fungsi tombol "Buang"	Black Box
Fungsi tombol "Nilai"	Black Box
Pengacakan Kartu	White Box
Artificial Intelligent	White Box

Pengujian Black Box

Pengujian *black box* yaitu pengujian yang berfokus pada persyaratan dan fungsionalitas dari sebuah perangkat lunak. Berikut ini akan dijabarkan beberapa pengujian yang dilakukan dengan menggunakan metode pengujian *black box*.

a) Fungsional Testing

Penanganan kesalahan mempunyai tujuan untuk memberikan informasi tentang kesalahan maupun penanganannya yang terjadi ketika proses sistem dijalankan. Berikut merupakan hasil pengujian dari beberapa fungsi yang terdapat di dalam aplikasi ini.

Tabel 4 Pengujian Tombol Main

Hasil Uji			
Data Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Klik tombol Main	Masuk ke halaman input nama pemain	Masuk ke halaman input nama pemain	berhasil

Tabel 5 Pengujian Input Nama Pemain

Hasil Uji (Benar)			
Data Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Nama Pemain	Pengisian data pada textbox berhasil	Data nama pemain benar	Berhasil
Klik tombol Mulai	Masuk ke halaman selanjutnya	Masuk ke halaman selanjutnya	Berhasil

Tabel 6 Pengujian Tombol acak

Hasil Uji (Benar)			
Data Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Klik tombol metode pengacakan	Masuk ke halaman permainan dengan metode pengacakan yang dipilih	Masuk ke halaman permainan dengan metode pengacakan yang dipilih	Berhasil

Tabel 6 Pengujian Tombol acak

Hasil Uji (Benar)			
Data Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Klik tombol metode pengacakan	Masuk ke halaman permainan dengan metode pengacakan yang dipilih	Masuk ke halaman permainan dengan metode pengacakan yang dipilih	Berhasil

Tabel 7 Pengujian Tombol Ambil

Hasil Uji (Benar)			
Proses	Yang Diharapkan	Pengamatan	Kesimpulan
Klik tombol ambil	Kartu ditumpukan paling atas diambil	Kartu ditumpukan diambil dan ditampilkan di slot kartu sementara	Berhasil

Tabel 8 Pengujian Tombol Nilai

Hasil Uji (Benar)			
Proses	Yang Diharapkan	Pengamatan	Kesimpulan
Klik tombol cek nilai	Sistem akan menghitung nilai pemain dan menampilkan poin pemain	Sistem menampilkan poin yang sesuai	Berhasil

Tabel 9 Pengujian Tombol Buang

Hasil Uji (Benar)			
Proses	Yang Diharapkan	Pengamatan	Kesimpulan
Klik pada kartu yang akan dibuang	Kartu yang akan dibuang terpilih ditandai dengan munculnya frame pada kartu	Pada kartu yang dipilih muncul frame	Berhasil
Klik tombol buang	Kartu yang telah dipilih / ditandai berpindah ke tumpukan kartu buangan	Kartu yang dipilih berpindah ke tumpukan kartu buangan	Berhasil

b) Pengujian Sistem

Berikut akan dijelaskan tentang hasil dari pengujian sistem yang telah dilakukan terhadap beberapa fungsi di dalam aplikasi ini.

Tabel 10 Hasil pengujian system

No.	Nama Pengujian	Hasil yang diharapkan	Hasil Pengujian	Status
1	Pengacakan Kartu	Index 52 kartu remi dapat teracak	Semua berjalan baik dan index kartu teracak sesuai dengan fungsi yang diharapkan	Berhasil
2.	Artificial Intelligent	Artificial Intelligent permainan kartu Remi 41 dapat berjalan dengan baik	Semua berjalan baik dan pemain dapat bermain melawan komputer	berhasil

 ISSN : 1858 – 3938

Pengujian White Box

Pengujian dengan metode white box adalah pengujian yang dilakukan berdasarkan fungsi-fungsi yang ada di dalam setiap form.
Penjelasannya adalah sebagai berikut :

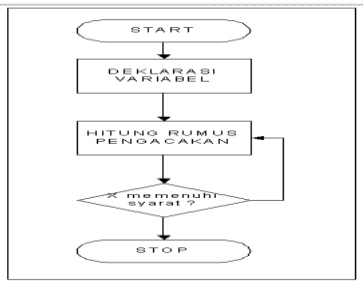
a) Pengujian Fungsi pengacakan kartu

Algoritma pengacakan kartu digunakan untuk mengacak kartu ketika pemain memilih salah satu metode pengacakan Diawali dengan pendeklarasian integer, lalu kemudian dilakukan pengecekan kondisi secara berulang dan penghitungan rumus berdasarkan metode Linear Congruential Generator (LCG).

Proses	Keterangan

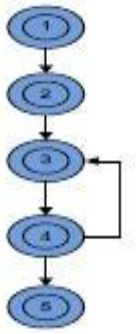
initComponents();
int a = 13;
int b = 7;
int m = 52;
int xn = 1;
int x;
int y;
int k = 0;
int r = 1;
for (x = 1 ; x < 53 ; x++){
y = ((a*(xn-1)+b)%51);
xn++;
}
	Rumus pembangkitan angka random menggunakan metode Linear Congruential generator

a. Basis Path Testing
Dalam melakukan pengujian ini, digunakan metode *graph*. Dimana langkah awal dari metode ini dibuatkan *flowchart* dari penggalan baris kode diats, adapun *flowchart*-nya adalah sebagai berikut :



Gambar 20 Flowchart Pengacakan Kartu

Pada setiap rancangan prosedural dapat diterjemahkan ke dalam *flow graph.* Dibawah ini merupakan bagian dari PDL (*Program Design Language*) dan *flowgraph*-nya. Ketika kondisi gabungan ditemukan, maka penggambaran flow graph akan menjadi lebih rumit. Kondisi gabungan biasanya muncul jika satu atau lebih operator boolean (OR, AND, NAND, NOR) ditemukan dalam perintah, seperti terlihat pada gambar dibawah ini :



Gambar 21 Flowgraph pengacakan kartu

b. Cyclomatic Complexity
Cyclomatic Complexity merupakan suatu sistem pengukuran yang menyediakan menyediakan ukuran kuantitatif dari kompleksitas logika suatu program. Pada basis path testing, hasil dari cyclomatic complexity menentukan banyaknya independent path. Independent path adalah sebuah kondisi pada program yang menghubungkan node awal dengan node akhir. Independent path adalah adalah sebuah kondisi pada program yang menghubungkan node awal dengan node akhir. Dari flowgraph di atas terlihat bahwa kode-kode yang telah dinotasikan dalam bentuk node memiliki 5 node, 5 edge dan 1 predicate, adapun susunannya adalah sebagai berikut :

1) Node = 1; 2; 3; 4; 5
2) Edge = 1→2; 2→3; 3→4; 4→3; 4→5
3) Predicate = 1

Dari hasil flowgraph diatas, maka dapat dilakukan penghitungan kompleksitas siklomatis dengan 2 persamaan yang digunakan yaitu :

1) Penghitungan berdasarkan lintasan dalam flowgraph atau path (p) dihasilkan :
P1 = 1→2→3→4→5
P2 = 1→2→3→4→3→4→5
2) Penghitungan berdasarkan pada rumus V(G) = E-N+2, dimana E menyatakan Edge dan N menyatakan Node, dihasilkan :
V(G) = E – N + 2
= 5 – 5 + 2

 ISSN : 1858 – 3938

3) Penghitungan berdasarkan rumus V(G)= P+1 menyatakan predicate, dihasilkan :

V(G) = P + 1
= 1 + 1
= 2

c. Graph Matriks

Metode selanjutnya yang dipakai untuk melakukan testing adalah menggunakan metode graf matriks,dimana pengisian nilai-nilai di dalam matriks didasarkan pada hubungan antar node yang ada di dalam flowgraph. Jika ada hubungan antara node maka diberikan nilai 1, dan jika tidak ada hubungan maka diberi nilai 0 (nol). Hasil akhir dari metode graf matriks ini adalah penjumlahan dari hubungan antar node tersebut, hal ini dapat dilihat dalam tabel berikut :

Tabel 11 Graf Matriks Pengacakan Kartu

	1	2	3	4	5	
1		1				1 – 1 = 0
2			1			1 – 1 = 0
3				1		1 – 1 = 0
4			1		1	2 – 1 = 1
5						
						1 + 1 = 2

b) Pengujian Fungsi Artificial Intelligent

Algoritma permainan komputer menggunakan artificial intelligent digunakan untuk proses bermain oleh komputer. Diawali dengan pendeklarasian string dan integer yang digunakan lalu kemudian dilakukan pemanggilan method yang sesuai dengan kondisi permainan yang sedang berjalan.

Proses	Keterangan

public void nowPlaying(int player){
player = 1;
if(player == 0){
}else{
Random ran = new Random();
int stat;
stat = ran.nextInt(cardList.size());
ImageIcon icon;
icon = new ImageIcon("src/newfinal/gambar/card/"+ cardList.get(stat)+"");
kartuSementara.setIcon(icon);
calonKartuUser = cardList.get(stat);
kS = cardList.get(stat);
kS = getNameCard(kS);
cardList.remove(stat);
icon = null;
if (cardList.size()==0){
JOptionPane.showMessageDialog (null, "kartu habis");
ImageIcon icon2;
icon2 = new ImageIcon("src/newfinal/gambar/backside.jpg");
kartuSementara.setIcon(icon2);
cekNilai(1);
//cekNilai(0);
} } }
	Atificial intelligent dimulai dengan pemanggilan method untuk permainan oleh komputer. Diawali dengan pembagian kartu yang diperoleh oleh komputer

Proses	Keterangan

private void mainKomputer(int a) {
ImageIcon icon;
if (a==0) {
}
if (a ==1) {
String kartuKeluar = cekKartuEx();
int p = cekPosisiKartu(kartuKeluar);
if (p == 1) {
lblAI01.seticon(kartuSementara.geticon());
}
if (p == 2) {
lblAI02.seticon(kartuSementara.geticon());
}
if (p == 3) {
lblAI03.seticon(kartuSementara.geticon());
}
if (p == 4) {
lblAI04.seticon(kartuSementara.geticon());
}
icon = new ImageIcon("src/newfinal/gambar/backside.jpg");
kartuSementara.seticon(icon);
}
if (a ==2) {
String kartuKeluar = kondisi2();
if (kartuKeluar != null) {
icon = new ImageIcon("src/newfinal/gambar/"+kartuKeluar+".jpg");
trashCard.seticon(icon);
int p = cekPosisiKartu(kartuKeluar);
if (p == 1) {
lblAI01.seticon(kartuSementara.geticon());
}
if (p == 2) {
lblAI02.seticon(kartuSementara.geticon());
}
if (p == 3) {
lblAI03.seticon(kartuSementara.geticon());
}
if (p == 4) {
lblAI04.seticon(kartuSementara.geticon());
}
icon = new ImageIcon("src/newfinal/gambar/backside.jpg");
kartuSementara.seticon(icon);
} else {
}
}
}
	Method main komputer dijalanankan ketika permainan telah dimulai, ditandai dengan set icon kartu komputer dengan kartu yang dimainkan.

Proses	Keterangan

private String kondisi2() {
String[] cocok; cocok = new String[5];
int[] ntCocok;
ntCocok = new int[5];
String[] pointC;
pointC = new String[5];
String[] beda;
beda = new String[5];
int[] ntBeda;
ntBeda = new int[5];
String[] pointB;
pointB = new String[5];
int it = 0; int it2=0;
Class_Nilai_Kartu objCNK = new Class_Nilai_Kartu();
String a = objCNK.jenisKartu(kS);
int na = objCNK.getNilai(kS);
String b = objCNK.jenisKartu(lA1);
int nb = objCNK.getNilai(lA1);
String c = objCNK.jenisKartu(lA2);
int nc = objCNK.getNilai(lA2);
String d = objCNK.jenisKartu(lA3);
int nd = objCNK.getNilai(lA3);
String f = objCNK.jenisKartu(lA4);
int nf = objCNK.getNilai(lA4);
if (a.equals(b)) {
cocok[it] = a; ntCocok[it] = na; pointC[it] = kS; it++;
cocok[it] = b; ntCocok[it] = nb; pointC[it] = lA1; it++;
} else {
beda[it2] = b; ntBeda[it2] = nb; pointB[it] = lA1; it2++;
}
if (a.equals(c)) {
cocok[it] = c; ntCocok[it] = nc; pointC[it] = lA2; it++;
} else {
beda[it2] = c; ntBeda[it2] = nc; pointB[it] = lA2; it2++;
}
if (a.equals(d)) {
cocok[it] = d; ntCocok[it] = nd; pointC[it] = lA3; it++;
} else {
beda[it2] = d; ntBeda[it2] = nd; pointB[it] = lA3; it2++;
}
if (a.equals(f)) {
cocok[it] = f; ntCocok[it] = nf; pointC[it] = lA4; it++;
} else {
beda[it2] = f; ntBeda[it2] = nf; pointB[it] = lA4; it2++;
}
// htenghitung nilai array cocok dan beda
String kept = "";
int nilaiCocok=0; int nilaiBeda=0;
for (int i=0;i<it;i++){
nilaiCocok = nilaiCocok + ntCocok[i];
}
for (int i=0;i<=it2;i++){
nilaiBeda = nilaiBeda + ntBeda[i];
}
// Ambil keputusan untuk menentukan array -> di buang kartu
if (nilaiCocok >= nilaiBeda) {
kept = "beda";
} else {
kept = "cocok";
}
	Method kondisi2 dipanggil ketika komputer menjalankan giliran, kemudian melakukan pemeriksaan kartu yang dipegang. Apabila kartu yang dipegang terdapat dua jenis kartu yang sama, maka akan dipanggil method kondisi2. Dimana method tersebut akan melakukan pengecekan nilai tertinggi dari masing-masing jenis, jenis kartu dengan nilai total terendah akan dibuang salah satunya. Kartu yang dibuang adalah yang bernilai paling besar. Untuk menentukan kartu yang total jenisnya paling rendah dan kartu yang bernilai paling besar digunakan algoritma sorting

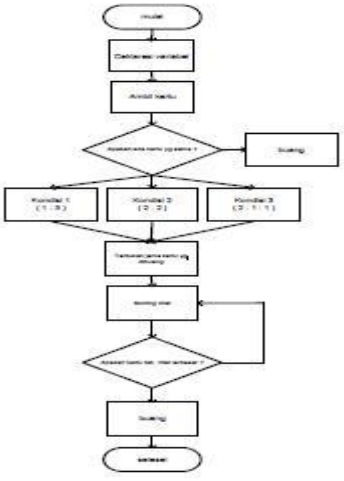
 ISSN : 1858 – 3938

```
String BuangKartu = "";
if (kept.equals("beda")) {
  // Sorting, terbesar di buang
  int temp; int posB=5; String tempPos;
  // Algoritma Sorting
  for (int i = 0; i< it; i++) {
    for (int j =(it - 1);j>=(i+1) ;j--) {
      if (ntBeda[j] < ntBeda[j-1] ) {
        temp = ntBeda[j];
        ntBeda[j] = ntBeda[j-1];
        ntBeda[j-1]=temp;

        tempPos = pointB[j];
        pointB[j] = pointB[j-1];
        pointB[j-1]=tempPos;
      } }
    if (pointB[i] != "") {
      posB = i;
    } }
  BuangKartu = pointB[posB];
  } else {
  // Sorting, terbesar di buang
  int temp; int posC=5; String tempPos;
   // Algoritma Sorting
  for (int i = 0; i< it2; i++) {
  for (int j =(it2 - 1);j>=(i+1) ;j--) {
      if (ntCocok[j] < ntCocok[j-1] ) {
        temp = ntCocok[j];
        ntCocok[j] = ntCocok[j-1];
        ntCocok[j-1]=temp;
        tempPos = pointC[j];
        pointC[j] = pointC[j-1];
        pointC[j-1]=tempPos;
      } }
    if (pointC[i] != "") {
      posC = i;
    } }
  BuangKartu = pointB[posC];
```

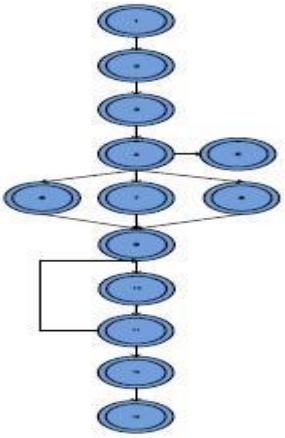
a. Basis Path Testing

Dalam melakukan pengujian ini, digunakan metode *graph*. Dimana langkah awal dari metode ini dibuatkan *flowchart* dari penggalan baris kode diatas, adapun *flowchart*-nya adalah sebagai berikut :



Gambar 22 Graph Flowchart Artificial Intelligent

Pada setiap rancangan prosedural dapat diterjemahkan ke dalam *flow graph.* Dibawah ini merupakan bagian dari PDL (*Program Design Language*) dan *flowgraph*-nya. Ketika kondisi gabungan ditemukan, maka penggambaran flow graph akan menjadi lebih rumit. Kondisi gabungan biasanya muncul jika satu atau lebih operator boolean (OR, AND, NAND, NOR) ditemukan dalam perintah, seperti terlihat pada gambar dibawah ini:



Gambar 23 *Flowgraph* Artificial Intelligent

b. Cyclomatic Complexity

Cyclomatic Complexity merupakan suatu sistem pengukuran yang menyediakan menyediakan ukuran kuantitatif dari kompleksitas logika suatu program. Pada basis path testing, hasil dari cyclomatic complexity menentukan banyaknya independent path.

DAFTAR PUSTAKA

[KDR 04] Kadir, Abdul.2004. DASAR PEMROGRAMAN JAVA 2, Penerbit : Andi, Yogyakarta.

[MUN 05] Munawar. 2005. PEMODELAN VISUAL DENGAN UML, Penerbit : Graha Ilmu, Yogyakarta.

[MUN 06] Munir, Rinaldi.2006. KRIPTOGRAFI, Penerbit : Informatika, Bandung.

[PET 10] Petri, Jurgen. 2010. *NETBEANS PLATFORM 6.9 DEVELOPER'S GUIDE,* Packt Publishing Ltd., Birmingham, B27 6PA, UK

[PRE 10] Pressman, Roger S. 2010. *REKAYASA PERANGKAT LUNAK PENDEKATAN,* Penerbit : Andi, Yogyakarta.

[WOL 03] Wolf, Mark J.P & Perron,Bernard, eds.2003. *THE VIDEO GAME THEORY,* Reader. Routledge. _____ ehow.com/how_4481285_play-card-game-41.html