KALBIScientia
Jurnal Sains dan Teknologi

ISSN 2356 - 4393

# Aplikasi Solusi Permainan Solitaire Menggunakan Algoritma Greedy

**Halim Agung[1)], Cindy Fransisca [2)], Johanes Fernandes Andry[3)]**

[1), 2)] Teknik Informatika, Fakultas Teknologi dan Desain, Universitas Bunda Mulia
Jalan Lodan Raya No. 2 Ancol, Jakarta Utara 14430
*Email: hagung@bundamulia.ac.id*

[3)] Sistem Informasi, Fakultas Teknologi dan Desain, Universitas Bunda Mulia
Jalan Lodan Raya No. 2 Ancol, Jakarta Utara 14430
*Email: jandry@bundamullia.ac.id*

***Abstract:*** *Solitaire is one game that is easy to learn but difficult to advanced because of the many steps that can be taken transfer of cards. This application aims to be able to learn the game of solitaire repeatedly and make users more quickly understand how to operate a game of solitaire with properly. The algorithm can be used is a greedy algorithm. Implementation of greedy algorithm in a solitaire game that can be used as an application tutorial of solitaire game that provides measures of any displacement of solitaire card arrangement that is still random until all cards are stacked neatly on the foundation in accordance with the rules of solitaire game on generally. The methodhology in the design and implementation of these applications is the System Development Life Cycle (SDLC). SDLC is used waterfall. The conclusion of this study is a greedy algorithm will seek the path of the card will be moved, because when the game starts, greedy algorithm prepare appropriately exit the path card so that the game can be completed.*

***Keywords:*** *artificial intelligent, greedy, solitaire*

***Abstrak:*** *Solitaire merupakan salah satu permainan yang mudah dipelajari tetapi sulit dikuasai karena banyaknya langkah perpindahan kartu yang dapat diambil. Aplikasi ini bertujuan agar dapat mempelajari permainan solitaire secara berulang-ulang dan membuat user lebih cepat mengerti cara mengoperasikan permainan solitaire dengan benar. Algoritma yang digunakan dalam penelitian ini adalah algoritma greedy. Implementasi Algoritma Greedy dalam permainan solitaire memberikan langkah-langkah dari setiap perpindahan kartu solitaire dari susunan kartu yang masih random sampai setiap kartu tersusun rapi di foundation kartu sesuai dengan aturan permainan solitaire pada umumnya. Metode yang digunakan dalam perancangan dan implementasi aplikasi ini adalah System Development Life Cycle (SDLC). SDLC yang digunakan adalah waterfall. Kesimpulan penelitian ini adalah algoritma greedy akan mencari jalur dari kartu yang akan dipindahkan, karena pada saat permainan di mulai, algoritma greedy menyiapkan jalur keluarnya kartu secara tepat agar permainan dapat diselesaikan.*

***Kata kunci:*** *kecerdasan buatan, greedy, solitaire.*

## I. PENDAHULUAN

Solitaire adalah salah satu permainan yang mudah untuk dipelajari tetapi sulit untuk dikuasai karena banyaknya langkah yang dapat diambil. Permainan atau game biasanya memiliki karakter yang dikontrol oleh user, dan karakter lawan yang dikontrol oleh game itu sendiri. Dimana kita harus merancang aturan-aturan yang nantinya akan dikerjakan oleh karakter lawan. Game akan menjadi menarik apabila karakter lawan (non-player) bereaksi dengan baik terhadap apa yang dilakukan oleh player. Hal ini akan memancing penasaran user dan membuat game menarik untuk dimainkan. Tujuan intinya adalah membuat non-player memiliki strategi yang cerdas untuk mengalahkan player. Pada bidang ini, AI dibutuhkan, yaitu untuk merancang dan menghasilkan game yang fun serta antarmuka antara man-machine yang cerdas dan menarik untuk dimainkan [1].

Untuk mempelajari permainan Solitaire ini, kecerdasan buatan (*Artificial Intelligence*) dapat membantu menyelesaikan permainan solitaire dengan kurun waktu lebih singkat daripada dimainkan oleh manusia [2]. Kata intelligence berasal dari bahasa

Latin intelligo yang berarti 'saya paham'. Jadi, dasar dari intelligence adalah kemampuan memahami dan melakukan aksi. Sebenarnya, area kecerdasan Batan (Artificial Intelligence) atau disingkat dengan AI, bermula dari kemunculan komputer sekitar tahun 1940-an, meskipun sejarah perkembangannya dapat dilacak hingga zaman Mesir kuno. Pada masa sekarang, perhatian difokuskan pada kemampuan komputer untuk mengerjakan sesuatu yang dapat dilakukan oleh manusia. Dalam hal ini, komputer tersebut dapat meniru kemampuan kecerdasan dan perilaku manusia [3].

McCarthy mendefinisikan sebagai, "AI merupakan cabang dari ilmu komputer yang berfokus pada pengembangan komputer untuk dapat memiliki kemampuan dan berperilaku seperti manusia". Untuk menguji definisi tersebut, Anda dapat membayangkan sekelompok robot yang berjalan dan bergerak dengan berbagai macam manuver, namun tidak menabrak satu sama lain [4].

Salah satu ilmu dari Artificial Intelligence adalah Natural Language Programming (NLP). NLP mempelajari bagaimana bahasa alami itu diolah sedemikian hingga user dapat berkomunikasi dengan komputer. Konsentrasi ilmu ini adalah interaksi antara komputer dengan bahasa natural yang digunakan manusia, yakni bagaimana komputer melakukan ekstraksi informasi dari input yang berupa natural language dan atau menghasilkan output yang juga berupa natural language, misalnya pada sistem Automated Online Assistant seperti Gambar 1 dan deteksi email spam yang cerdas [5].

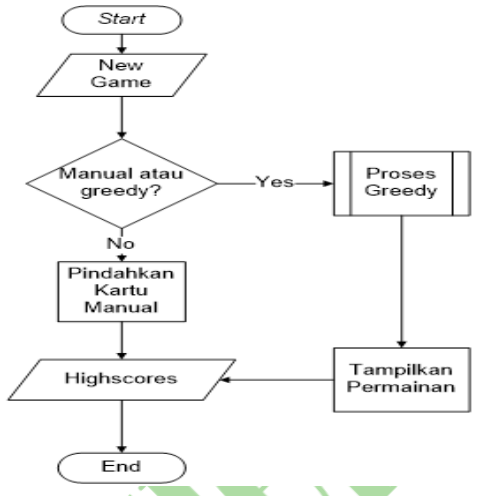
Salah satu algoritma yang dapat diterapkan di permainan Solitaire untuk mengambil langkah optimal yang dilakukan oleh sistem adalah algoritma greedy. Greedy yang berarti rakus tau tamak, prinsip dari greedy sendiri yaitu "take what get now" atau ambil yang kamu dapatkan sekarang dngan membentuk solusi langkah per langkah (step by step) dan pada setiap langkah akan dapat banyak pilihan untuk diksplorasi [6].

Tujuan penelitian adalah membangun aplikasi dengan algoritma greedy yang dapat mencari solusi dari permainan solitaire, sedangkan manfaatnya yaitu mengetahui langkah optimal yang di ambil oleh algoritma greedy dalam menyelesaikan permainan solitaire dari setiap langkahnya.

## II. METODE PENELITIAN

Untuk merancang dan mengimplementasikan algoritma Greedy pada aplikasi solitaire maka penulis menggunakan SDLC. Metode yang digunakan dalam SDLC adalah metode waterfall [7].

### A. Flow Permainan Solitaire




*Gambar 1. Flowchart Permainan Solitaire*

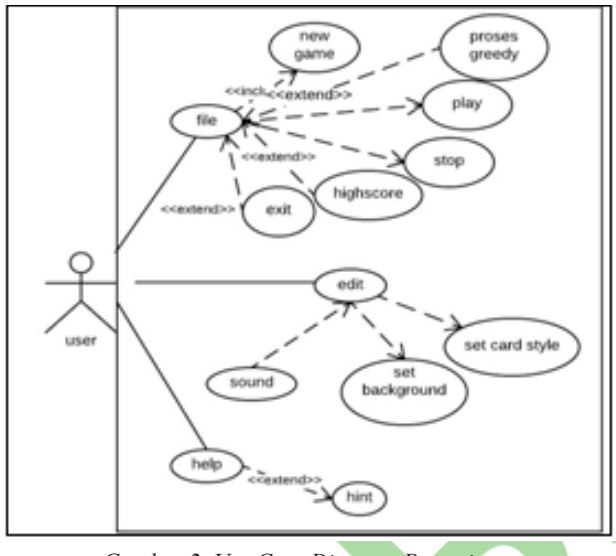
Pada Gambar 1 menjelaskan proses yang terjadi pada aplikasi solitaire. Dimulai dari penyediaan papan solitaire. Kemudian sistem otomatis melakukan *solving* permainan menggunakan algoritma greedy. Setelah sistem menyelesaikan pengacakan nilai kartu maka pemain dapat menyusun kartu yang telah di acak oleh sistem dari nilai kartu terbesar (king) sampai nilai kartu terkecil (ace). Setelah pemain menyelesaikan penyusunan kartu yang *random*, tumpukan kartu yang telah disusun ditempatkan pada empty deck agar kartu yang telah disusun tidak tercampur dengan kartu yang masih berantakan pada slot.

### B. Algoritma Greedy

Algoritma greedy akan menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara pada setiap langkahnya [5]. Nilai maksimum sementara ini dikenal dengan istilah local maximum. Pada kebanyakan kasus, algoritma greedy tidak akan menghasilkan solusi paling optimal, begitupun algoritma greedy biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup cepat. Algoritma greedy digunakan untuk memperoleh penyelesaian dari suatu permasalahan optimasi. Suatu permasalahan dengan n masukkan data dilakukan secara bertahap. Pertama dilakukan pemilihan solusi yang mungkin kemudian dari himpunan solusi yang mungkin tersebut akan diperoleh solusi optimal. Metode ini bekerja secara bertahap dengan memperhatikan setiap input data pada setiap keadaan. Pada setiap tahap, dibuat keputusan dengan memperhatikan ada atau tidak sebuah input data yang memberikan solusi optimal,

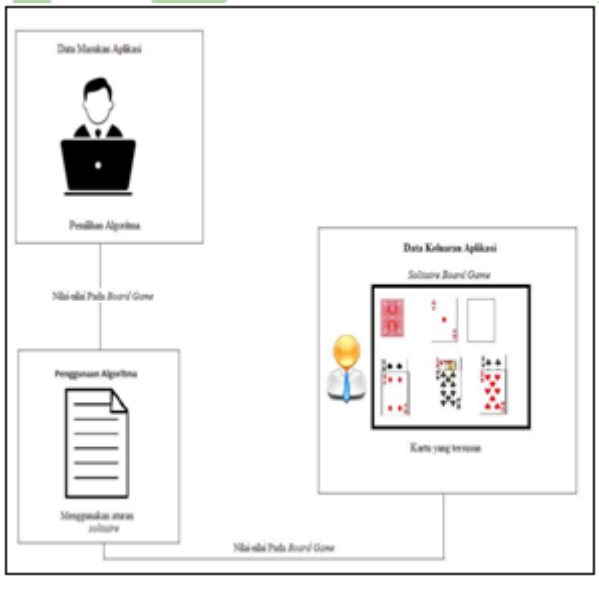
dan memperhatikan pula urutan data dalam proses pengambilannya.

Dalam aplikasi permainan solitaire dengan algoritma greedy, pemain dapat menyusun kartu dari nilai terbesar hingga terkecil dan ketika pemain tidak dapat melanjutkan permainan maka pemain dapat memilih proses greedy setelah menekan *button* greedy maka sistem akan menyelesaikan pemainan hingga menang.




*Gambar 2. Use Case Diagram Permainan Solitaire*

Pada Gambar 2 dijelaskan pemain dapat menyusun kartu dari nilai terbesar hingga terkecil dan ketika pemain tidak dapat melanjutkan permainan maka pemain dapat memilih proses greedy setelah menekan *button* greedy maka sistem akan menyelesaikan pemainan hingga menang.



*Gambar 3. Arsitektur Sistem*

Gambar 3 menjelaskan arsitektur sitem, yang menjelaskan ogram yang digunakan untuk menerima masukan berupa algoritma pada permainan solitaire yang akan menjalankan permainan solitaire. Sistem akan mengeksekusi algoritma yang dipakai untuk menjalankan permainan ini. Algoritma yang di pergunakan untuk menjalankan permainan adalah algoritma greedy. Keluaran dari aplikasi ini berupa permainan kartu solitaire yang telah tersusun rapi pada bagian foundation dari urutan ace (pada bagian bawah) sampai king (pada bagian top). Arsitektur sistem di jelaskan pada Gambar 3.

## III. HASIL DAN PEMBAHASAN

```
public void greedy()
{
    Card CHOOSEN_CARD = null;
    Card TMP_CARD = null;
    ArrayList<Card> CHOOSEN_MANY_CARD = new ArrayList<Card>();

    int fullCount = 0;
    int indeks, posisi, i, indeksCard = 0;
    boolean ketemu;
    boolean keluarIndex;

    int sizes = 0;
    int cardNotOpen = 0;

    resumeTimer();
    setGameStarted(true);
```

*Gambar 4.Function Greedy*

Gambar 4. berisi variabel-variabel lokal yang digunakan untuk melakukan eksekusi algoritma greedy. Berikut ini masing-masing penjelasannya: (1) Card CHOOSEN_CARD = null; CHOOSEN_CARD Merupakan variabel dengan tipe data Card yang digunakan sebagai penampung kartu yang akan dipindahkan; (2) Card TMP_CARD = null; TMP_CARD Merupakan variabel dengan tipe data Card yang digunakan sebagai penampung kartu yang akan dipindahkan dalam proses foundation to solitaire stack to solitaire stack; (3) ArrayList<Card> CHOOSENMANY_CARD=newArrayList(Card>(); CHOOSEN_MANY_CARD adalah variabel dengan tipe data Collection yaitu ArrayList yang mengacu kepada class Card. Tipe data ini digunakan karena ukuran datanya tidak dapat ditentukan. Variabel ini digunakan untuk memindahkan banyak kartu sekaligus dalam proses perpindahan kartu Many Card Solitaire Stack to One Card Solitaire Stack; (4) Int fullCount = 0; fullCount adalah variable dengan tipe int (integer) yang digunakan untuk menghitung berapa banyak kartu yang sudah penuh di foundation; (5) Int indeks, posisi, i, indeksCard = 0; Indeks, posisi, i, indeksCard merupakan tipe data int (integer) dengan fungsi sebagai berikut: (a) Indeks

Digunakan untuk menyeleksi kartu mana yang akan dipindahkan berdasarkan posisinya; (b) Posisi Digunakan sebagai variabel pengulang untuk mengetahui posisi kartu yang tepat dengan variabel indeksnya; (c) I Digunakan sebagai variabel pengulang. IndeksCard Digunakan untuk mengetahui berapa banyak kartu yang tersisa di tumpukan Solitaire Stack; dan (d) Boolean Ketemu Ketemu adalah variabel dengan tipe data boolean yang digunakan untuk mengecek apakah ada kartu yang bisa dipindahkan. Jika ada, maka variabel ketemu bernilai true, sebaliknya false; (6) Boolean KeluarIndex KeluarIndex merupakan tipe data boolean Digunakan untuk mengetahui apakah Dealt Stack sudah kosong atau belum. Jika kosong, maka nilai keluarIndex = true, sebaliknya nilai keluarIndex = false; (7) Int sizes = 0; Sizes merupakan tipe data int (integer) yang digunakan untuk mengetahui berapa banyak kartu yang terbuka di tumpukan Solitaire Stack; (8) Int cardNotOpen = 0; cardNotOpen adalah tipe data int (integer) yang digunakan untuk mengetahui berapa banyak kartu yang tertutup di tumpukan Solitaire Stack; (9) ResumeTimer(); ResumeTime(); merupakan sebuah function yang digunakan untuk melanjutkan waktu permainan; (10) SetGameStarted(true); SetGameStarted(true); merupakan sebuah function yang digunakan untuk menandai suatu permainan itu dimulai; dan (11) Algoritma greedy dimulai dengan pengecekan variabel fullCount dengan kondisi saat nilai variabel fullCount != (tidak sama dengan) 4 (karena ada 4 foundation), maka perulangan while akan berjalan dan akan melakukan ekseskusi intruksi program didalamnya. Sebaliknya perulangan while akan berakhir dan algoritma greedy berakhir. Jika nilai variabel fullCount tidak sama dengan 4 maka langkah pertama program akan melakukan eksekusi Thread yang berfungsi untuk memperlambat proses perulangan sehingga user dapat melihat pergerakan kartu. Selanjutnya, program akan melakukan inisialisasi variabel posisi = 0 dan indeks dengan nilai random (acak) antara 0 sampai dengan 6 (karena nilai variabel SOLITAIRE_STACK_COUNT memiliki nilai 7 dan tidak dapat diubah). Kemudian program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks.

Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SolitaireStack (karena class SolitareStack, DealtCardsStack dan Foundation turunan dari class Stack), maka akan dilakukan pengecekan antara variabel posisi dan indeks. Sebaliknya, program akan melanjutkan perulangan for. Jika nilai variabel posisi == (sama dengan) dengan nilai variabel indeks, maka program akan melakukan proses untuk memperoleh berapa banyak kartu yang ada di tumpukan tersebut dengan statement stack.getAllCards().size(); Sebaliknya nilai dari variabel posisi akan dinaikkan sebanyak 1 (satu) dengan statement posisi++. Nilai dari statement stack.getAllCards().size() tersebut akan ditampung ke dalam variabel indeksCard. Jika nilai indeksCard tidak memiliki nilai 0 (artinya masih ada tumpukan), maka program akan melakukan pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack.getTopCard(); dan nilai dari statement ini akan masuk ke dalam variabel CHOOSEN¬_CARD. Karena variabel CHOOSEN_CARD sudah diinisialisasi maka perulangan for harus diberhentikan dengan statement break.

Kondisi jika kartu akan bergeser ketempat lain. Program mulai dengan inisialisasi variabel ketemu=false. Kemudian program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Perulangan for ini, program akan melakukan pengecekan jika variabel stackmengandung (instanceof) tipe data class foundation, maka akan dilakukan pengecekan apakah variabel indeks Card tidak memiliki nilai 0 (Jika 0 berarti tidak ada kartu yang ditampung ke variabel CHOOSEN_CARD).

Jika nilai dari variabel indeksCard tidak sama dengan 0, maka program akan melakukan pengecekan kembali dengan isValidMove (CHOOSEN_CARD). Function ini dapat ditemukan pada class foundation dan memiliki nilai true dengan kriteria sebagai berikut: (1) Kartu yang dipindah adalah kartu Ace; dan (2) Kartu yang akan dipindahkan memiliki jenis kartu (clubs, diamonds, hearts, spades) yang sama dengan yang ada di foundation dan selisih antara kartu yang ada di foundation dengan kartu yang akan dipindah sama dengan 1. Selain dari kriteria diatas, program tidak akan melakukan eksekusi program didalam setelah dilakukan pengecekan function isValidMove. Jika pengecekan function isValidMove menghasilkan nilai true maka program akan melakukan perpindahan kartu dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya nilai data dari variabel cards tersebut akan ditambahkan ke dalam variabel stack yang saat ini berada di foundation. Selanjutnya program akan mengecek apakah nilai

dari variabel SOUND_ON memiliki nilai true. Jika variabel SOUND_ON memiliki nilai true, maka program akan mengeluarkan suara kartu untuk menandakan perpindahan kartu. Kemudian program akan melakukan inisialisasi variabel ketemu sama dengan true (berarti kartu berhasil dipindahkan) dan statement break; untuk menghentikan perulangan for.

Kondisi jika menggambarkan penghapusan kartu yang sudah dipindah ke foundation tetapi kartu tersebut masih ada di class SolitaireStack. Pertama tama, program akan melakukan pengecekan nilai variabel ketemu dahulu apakah nilai tersebut true atau false. Jika nilai variabel ketemu adalah true, maka terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah false, maka tidak terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah true, maka langkah selanjutnya program akan inisialisasi variabel posisi dengan nilai 0 dan akan dilakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya yang berdasarkan pada variabel stacks.

Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SoitaireStack, maka akan dilakukan pengecekan apakah variabel indeksCard tidak memiliki nilai 0 (Jika 0 berarti tidak ada kartu yang ditampung ke variabel CHOOSEN_CARD). Jika nilai dari variabel indeksCard tidak sama dengan 0, maka program akan melakukan pengecekan kembali apakah nilai variabel posisi sama dengan nilai variabel indeks. Jika nilai variabel posisi sama dengan nilai variabel indeks, maka instruksi program selanjutnya, yaitu penghapusan kartu yang akan dieksekusi.

Sebaliknya, program akan menaikkan nilai variabel posisi sebanyak 1 dan menuju perulangan berikutnya. Jika nilai variabel posisi sama dengan nilai variabel indeks, maka langkah selanjutnya adalah pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack.getTopCard(); dan nilai dari statement ini akan masuk ke dalam variabel CHOOSEN¬_CARD. Selanjutnya program akan melakukan penghapusan kartu dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya variabel stack yang saat ini di SolitaireStack akan mencari dan menghapus nilai data dari variabel cards tersebut. Selanjutnya function repaint(); akan dieksekusi agar program melakukan penggambaran ulang dari kartu yang dipindah dan dihapus.

Melakukan eksekusi Thread yang berfungsi untuk memperlambat proses perulangan sehingga user dapat melihat pergerakan kartu. Selanjutnya, program akan melakukan inisialisasi variabel posisi = 0 dan indeks dengan nilai random (acak) antara 0 sampai dengan 6 (karena nilai variabel SOLITAIRE_STACK_COUNT memiliki nilai 7 dan tidak dapat diubah kembali).

Kemudian program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SolitaireStack, maka akan dilakukan pengecekan antara variabel posisi dan indeks. Sebaliknya, program akan melanjutkan perulangan for. Jika nilai variabel posisi == (sama dengan) dengan nilai variabel indeks, maka program akan melakukan proses untuk memperoleh berapa banyak kartu yang ada di tumpukan tersebut dengan statement stack.getAllCards().size(); Sebaliknya nilai dari variabel posisi akan dinaikkan sebanyak 1 (satu) dengan statement posisi++. Nilai dari statement stack.getAllCards().size() tersebut akan ditampung ke dalam variabel indeksCard. Jika nilai indeksCard tidak memiliki nilai 0 (artinya masih ada tumpukan), maka program akan melakukan pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack.getTopCard(); dan nilai dari statement ini akan masuk ke dalam variabel CHOOSEN¬_CARD. Selanjutnya akan dilakukan pengecekan kembali jika nilai variabel CHOOSEN_CARD sama dengan 13 (King) maka nilai dari variabel indeksCard menjadi 0 karena karung King tidak boleh dipindah. Karena variabel CHOOSEN_CARD sudah diinisialisasi maka perulangan for harus diberhentikan dengan statement break.

Kondisi jika menggambarkan kartu bergeser ketempat lain. Program mulai dengan inisialisasi variabel ketemu = false. Kemudian program akan melakukan pengecekan apakah variabel indeksCard tidak memiliki nilai 0 (Jika 0 berarti tidak ada kartu yang ditampung ke variabel CHOOSEN_CARD). Jika nilai variabel indeksCard tidak memiliki nilai 0, maka perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks.

Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SolitaireStack, maka program akan melakukan pengecekan kembali dengan function isValidMove(CHOOSEN_CARD).

Function ini dapat ditemukan pada class SolitaireStack dan memiliki nilai true dengan kriteria sebagai berikut: (1) Kartu yang dipindah adalah kartu King dan tempat kartu yang akan ditempatkan itu kosong; dan (2) Kartu yang akan dipindahkan memiliki jenis kartu (clubs, diamonds, hearts, spades) yang berbeda dengan yang ada di tumpukan SolitaireStack yang lain dan selisih antara kartu yang ada di SolitaireStack dengan kartu yang akan dipindah sama dengan 1.

Selain dari kriteria diatas, program tidak akan melakukan eksekusi program didalam setelah dilakukan pengecekan function isValidMove. Jika pengecekan function isValidMove menghasilkan nilai true maka program akan melakukan perpindahan kartu dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya nilai data dari variabel cards tersebut akan ditambahkan ke dalam variabel stack yang saat ini berada di SolitaireStack. Selanjutnya program akan mengecek apakah nilai dari variabel SOUND_ON memiliki nilai true. Jika variabel SOUND_ON memiliki nilai true, maka program akan mengeluarkan suara kartu untuk menandakan perpindahan kartu. Kemudian program akan melakukan inisialisasi variabel ketemu sama dengan true (berarti kartu berhasil dipindahkan) dan statement break; untuk menghentikan perulangan for.

Kondisi jika menggambarkan penghapusan kartu yang sudah dipindah ke SolitaireStack yang baru tetapi kartu tersebut masih ada di class SolitaireStack awal. Pertama-tama, program akan melakukan pengecekan nilai variabel ketemu dahulu apakah nilai tersebut true atau false. Jika nilai variabel ketemu adalah true, maka terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah false, maka tidak terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah true, maka langkah selanjutnya program akan inisialisasi variabel posisi dengan nilai 0 dan akan dilakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SoitaireStack, maka akan dilakukan pengecekan apakah variabel indeksCard tidak memiliki nilai 0 (Jika 0 berarti tidak ada kartu yang ditampung ke variabel CHOOSEN_CARD). Jika nilai dari variabel indeksCard tidak sama dengan 0, maka program akan melakukan pengecekan kembali apakah nilai variabel posisi sama dengan nilai variabel indeks. Jika nilai variabel posisi sama dengan nilai variabel indeks, maka instruksi program selanjutnya, yaitu penghapusan kartu akan dieksekusi.

Sebaliknya, program akan menaikkan nilai variabel posisi sebanyak 1 dan menuju perulangan berikutnya. Jika nilai variabel posisi sama dengan nilai variabel indeks, maka langkah selanjutnya adalah pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack. getTopCard(); dan nilai dari statement ini akan masuk ke dalam variabel CHOOSEN¬_CARD. Selanjutnya program akan melakukan penghapusan kartu dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya variabel stack yang saat ini di SolitaireStack akan mencari dan menghapus nilai data dari variabel cards tersebut. Selanjutnya function repaint(); akan dieksekusi agar program melakukan penggambaran ulang dari kartu yang dipindah dan dihapus.

Gambar 5. dimulai dengan melakukan eksekusi Thread yang berfungsi untuk memperlambat proses perulangan sehingga user dapat melihat pergerakan kartu. Selanjutnya, program akan melakukan inisialisasi variabel ketemu sama dengan false, variabel posisi sama dengan 0, dan indeks dengan nilai random (acak) antara 0 sampai dengan 6 (karena nilai variabel SOLITAIRE_STACK_COUNT memiliki nilai 7 dan tidak dapat diubah).

Kemudian program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SolitaireStack, maka akan dilakukan pengecekan antara variabel posisi dan indeks. Sebaliknya, program akan melanjutkan perulangan for.

Jika nilai variabel posisi tidak sama dengan nilai variabel indeks maka program akan menaikkan nilai variabel posisi sebanyak 1. Sebaliknya program akan melakukan inisialisasi nilai variabel cardNotOpen sama dengan 0 dan nilai variabel cardNotOpen akan diinisialisasi dengan function getIsNotTurnedCards(). size() untuk memperoleh berapa banyak kartu yang belum dibuka. Bila function getIsNotTurnedCards(). size() menghasilkan nilai null maka cardNotOpen akan diinisialisasi dengan nilai 0, sebaliknya nilai dari function getIsNotTurnedCards().size() akan masuk kedalam variabel cardNotOpen.

Jika nilai variabel cardNotOpen sama dengan 0 (artinya semua kartu sudah terbuka di tumpukan

tersebut) maka program akan melakukan inisialisasi variabel sizes dengan function getAvailableCards(). size(); (Berfungsi untuk mengambil berapa banyak kartu yang tersisa di tumpukan tersebut).

Jika function getAvailableCards().size(); menghasilkan nilai null maka tumpukan tersebut sudah tidak memiliki kartu dan variabel sizes akan diinisialisasi dengan nilai 0. Sebaliknya, nilai sizes akan diinisialisasi dengan function getAvailableCards(). size(); Jika variabel sizes memiliki nilai tidak sama dengan 0 (berarti masih ada kartu di kartu tersebut) maka program akan melakukan pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack.getTopCard(); dan nilai dari statement ini akan masuk ke dalam variabel TMP¬_CARD.

Berikutnya program akan melakukan pengecekan apakah nilai dari variabel TMP_CARD bukan kartu ash. Jika nilai dari variabel TMP_CARD adalah kartu ash maka variabel sizes akan diinisialisasi dengan nilai sama dengan 0 (Karena kartu ash tidak perlu dipindahkan). Sebaliknya variabel CHOOSEN_ CARD akan diinisialisasi dengan function getBottomAvailableCard(sizes); yang berfungsi untuk mengambil kartu sebelum kartu yang tertutup. Variabel CHOOSEN_MANY_CARD juga akan diinisialiasi dengan function getAvailableCard(); yang berfungsi untuk mengambil semua kartu sebelum kartu yang ditutup. Jika nilai dari variabel CHOOSEN_CARD memiliki nilai sama dengan 13 (King), maka variabel sizes akan diinisialisasi dengan nilai sama dengan 0 (karena kartu King tidak pindah). Jika nilai variabel cardNotOpen tidak sama dengan 0 (artinya ada kartu yang belum terbuka di tumpukan tersebut) maka program akan melakukan inisialisasi variabel sizes dengan function getAvailableCards(). size(); (Berfungsi untuk mengambil berapa banyak kartu yang tersisa di tumpukan tersebut). Jika function getAvailableCards().size(); menghasilkan nilai null maka tumpukan tersebut sudah tidak memiliki kartu dan variabel sizes akan diinisialisasi dengan nilai 0. Sebaliknya, nilai sizes akan diinisialisasi dengan function getAvailableCards().size(); Jika variabel sizes memiliki nilai tidak sama dengan 0 (berarti masih ada kartu di kartu tersebut) maka program akan melakukan pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack. getTopCard(); dan nilai dari statement ini akan masuk ke dalam variabel TMP-_CARD.

Berikutnya program akan melakukan pengecekan apakah nilai dari variabel TMP_CARD bukan kartu ash. Jika nilai dari variabel TMP_ CARD adalah kartu ash maka variabel sizes akan

```
try{
    Thread.sleep(timer);
}
catch(Exception e){
    e.printStackTrace();
}

//-Many Cards-Solitaire Stack to One Card-Solitaire Stack-

ketemu = false;
posisi = 0;
indeks = (int) (Math.random() * SOLITAIRE_STACK_COUNT);

for(Stack stack : stacks)
{
    if(stack instanceof SolitaireStack){

        if( posisi == indeks)
        {
            cardNotOpen = 0;

            try
            {
                cardNotOpen = stack.getIsNotTurnedCards().size();
            }
            catch(NullPointerException u)
            {
                cardNotOpen = 0;
            }

            if( cardNotOpen == 0 )
            {
                try
                {
                    sizes = stack.getAvailableCards().size();
                }
                catch(NullPointerException n)
                {
                    sizes = 0;
                }

                if( sizes != 0 )
                {
                    TMP_CARD = stack.getTopCard();

                    if( !stack.isAshCard(TMP_CARD))
                    {
                        CHOOSEN_CARD = stack.getBottomAvailableCard(sizes);
                        CHOOSEN_MANY_CARD = stack.getAvailableCards();

                        if(CHOOSEN_CARD.getNumber() == 13)
                        {
                            sizes = 0;
                        }
                    }
                    else
                        sizes = 0;
                }

            }
            else
            {
                try
                {
                    sizes = stack.getAvailableCards().size();
                }
                catch(NullPointerException n)
                {
                    sizes = 0;
                }

                if( sizes != 0 )
                {
                    TMP_CARD = stack.getTopCard();

                    if( !stack.isAshCard(TMP_CARD))
                    {
                        CHOOSEN_CARD = stack.getBottomAvailableCard(sizes);
                        CHOOSEN_MANY_CARD = stack.getAvailableCards();
                    }
                    else
                        sizes = 0;
                }
            }
            break;
        }
        posisi++;
    }
}
```

*Gambar 5.Many Solitaire Stack to One Solitaire Stack-1*

diinisialisasi dengan nilai sama dengan 0 (Karena kartu ash tidak perlu dipindahkan). Sebaliknya variabel CHOOSEN_CARD akan diinisialisasi dengan function getBottomAvailableCard(sizes); yang berfungsi untuk mengambil kartu sebelum kartu yang tertutup. Variabel CHOOSEN_MANY_ CARD juga akan diinisialiasi dengan function getAvailableCard(); yang berfungsi untuk mengambil semua kartu sebelum kartu yang ditutup.

Kondisi jika menggambarkan kartu bergeser ketempat lain. Program mulai dengan inisialisasi variabel posisi sama dengan 0. Kemudian program akan melakukan pengecekan apakah nilai variabel sizes sama dengan 0. Jika nilai variabel sizes sama dengan 0, program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SolitaireStack, maka program akan dilakukan pengecekan antara variabel posisi dan indeks.

Sebaliknya, program akan melanjutkan perulangan for. Jika nilai variabel posisi tidak sama dengan nilai variabel indeks, maka program akan melakukan proses untuk memperoleh berapa banyak kartu yang ada di tumpukan tersebut dengan statement stack.getAllCards().size(); Nilai dari statement stack. getAllCards().size() tersebut akan ditampung ke dalam variabel indeksCard.

Selanjutnya, program akan melakukan pengecekan melakukan pengecekan dengan function isValidMove(CHOOSEN_CARD). Jika function isValidMove(CHOOSEN_CARD) memiliki nilai true, maka program akan melakukan perulangan for dengan menggunakan variabel c dengan tipe data class Card dan ukuran datanya berdasarkan variabel CHOOSEN_MANY_CARD. Selanjutnya nilai data atau kartu dari variabel c tersebut akan ditambahkan ke dalam variabel stack yang saat ini berada di SolitaireStack dengan statement addCards(); Berikutnya program akan mengecek apakah nilai dari variabel SOUND_ON memiliki nilai true.

Jika variabel SOUND_ON memiliki nilai true, maka program akan mengeluarkan suara kartu untuk menandakan perpindahan kartu. Kemudian program akan melakukan inisialisasi variabel ketemu sama dengan true (berarti kartu berhasil dipindahkan) dan statement break; untuk menghentikan perulangan for.

Kondisi jika menggambarkan penghapusan kartu yang sudah dipindah ke SolitaireStack tetapi kartu tersebut masih ada di class SolitaireStack awal. Pertama-tama, program akan inisialisasi variabel posisi dengan nilai 0 dan program akan melakukan pengecekan nilai variabel ketemu dahulu apakah nilai tersebut true atau false. Jika nilai variabel ketemu adalah true, maka terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah false, maka tidak terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah true, maka akan dilakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks.

Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SoitaireStack, maka akan dilakukan pengecekan apakah nilai variabel posisi sama dengan nilai variabel indeks. Jika nilai variabel posisi sama dengan nilai variabel indeks, maka instruksi program selanjutnya, yaitu penghapusan kartu akan dieksekusi.

Sebaliknya, program akan melakukan menaikkan nilai variabel posisi sebanyak 1 dan menuju perulangan berikutnya. Jika nilai variabel posisi sama dengan nilai variabel indeks, maka program akan melakukan inisialisasi variabel sizes dengan nilai dari function getAvailableCards().size(); Jika nilai dari variabel sizes tidak memiliki nilai 0, maka langkah selanjutnya adalah pengambilan semua kartu sebelum kartu yang tertutup dari tumpukan SolitaireStack dengan statement stack.getAvailableCards(); dan nilai dari statement ini akan masuk ke dalam variabel CHOOSEN¬_MANY_CARD. Selanjutnya program akan melakukan penghapusan kartu dengan perulangan for dengan variabel c dimana variabel CHOOSEN_MANY_CARD akan ditambahkan ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Card dan selanjutnya variabel stack yang saat ini di SolitaireStack akan mencari dan menghapus nilai data dari variabel cards tersebut dengan function removeCards(); dan program akan mengakhiri perulangan for dengan variabel stack dengan statement break.

Selanjutnya function repaint(); akan dieksekusi agar program melakukan penggambaran ulang dari kartu yang dipindah dan dihapus. Inisialisasi nilai variabel ketemu sama dengan false dan program akan melakukan pengecekan apakah nilai variabel ketemu sama dengan false. Jika nilai variabel ketemu sama dengan false, maka program akan menambah kartu baru dari deck dengan statement dealtCards. addNewCardsFromDeck(CARDS_TO_DEAL_ COUNT); dan program akan inisialisasi nilai variabel keluarIndex dengan nilai false. Bila kartu yang sudah ditambahkan dari deck sudah habis maka program akan melakukan inisialisasi variabel keluar Index sama dengan true. Berikutnya program akan melakukan pengecekan jika deck sudah kosong, maka deck akan tampil kosong.

Selanjutnya akan dilakukan pengecekan apabila nilai variabel keluarIndex tidak sama dengan false maka program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class

Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class DealtCardsStack, maka program akan melakukan pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack.getTopCard(); dan nilai dari statement ini akan masuk ke dalam variabel CHOOSEN-_CARD. Kondisi jika menggambarkan kartu bergeser ketempat lain. Program mulai perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class foundation, maka program akan melakukan perpindahan kartu dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya nilai data dari variabel cards tersebut akan ditambahkan ke dalam variabel stack yang saat ini berada di foundation.

Selanjutnya program akan mengecek apakah nilai dari variabel SOUND_ON memiliki nilai true. Jika variabel SOUND_ON memiliki nilai true, maka program akan mengeluarkan suara kartu untuk menandakan perpindahan kartu. Kemudian program akan melakukan inisialisasi variabel ketemu sama dengan true (berarti kartu berhasil dipindahkan) dan statement break; untuk menghentikan perulangan for.

Kondisi jika menggambarkan penghapusan kartu yang sudah dipindah ke foundation tetapi kartu tersebut masih ada di class DealtCardsStack. Pertama-tama, program akan melakukan pengecekan nilai variabel ketemu dahulu apakah nilai tersebut true atau false. Jika nilai variabel ketemu adalah true, maka terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah false, maka tidak terjadi proses perpindahan kartu sebelumnya.

Jika nilai variabel ketemu adalah true, maka langkah selanjutnya program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class DealtCardsStack, maka penghapusan kartu akan dieksekusi dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya variabel stack yang saat ini di DealtCardsStack akan mencari dan menghapus nilai data dari variabel cards tersebut dengan statement stack.removeCards(cards); Selanjutnya function repaint();

Pengecekan nilai variabel ketemu adalah true (beradasarkan proses DealtCardsStack to Foundation sebelumnya). Jika nilai variabel ketemu adalah true, maka langkah selanjutnya adalah program akan menambah kartu baru dari deck dengan statement dealtCards.addNewCardsFromDeck(CARDS_TO_DEAL_COUNT); dan program akan inisialisasi nilai variabel keluarIndex dengan nilai false.

Bila kartu yang sudah ditambahkan dari deck sudah habis maka program akan melakukan inisialisasi variabel keluarIndex sama dengan true. Berikutnya program akan melakukan pengecekan jika deck sudah kosong, maka deck akan tampil kosong.

Selanjutnya akan dilakukan pengecekan apabila nilai variabel keluarIndex tidak sama dengan false maka program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class DealtCardsStack, maka program akan melakukan pengambilan kartu yang paling atas dari tumpukan SolitaireStack dengan statement stack.getTopCard(); nilai dari statement ini akan masuk ke dalam variabel CHOOSEN-_CARD.

Kondisi jika menggambarkan kartu bergeser ketempat lain. Program mulai dengan inisialisasi variabel ketemu sama dengan false dan akan dilakukan pengecekan apakah nilai variabel keluarIndeks tidak sama dengan false. Jika nilai variabel keluarIndeks tidak sama dengan false, maka program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks.

Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class SolitaireStack, maka program akan melakukan pengecekan apakah nilai variabel stack bukan kartu Ace. Jika nilai variabel stack bukan kartu ash maka program akan melakukan pengecekan kembali dengan function isValidMove(CHOOSEN_CARD). Jika pengecekan function isValidMove(CHOOSEN_CARD) menghasilkan nilai true maka program akan melakukan perpindahan kartu dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya nilai data dari variabel cards tersebut akan ditambahkan ke dalam

variabel stack yang saat ini berada di foundation. Selanjutnya program akan mengecek apakah nilai dari variabel SOUND_ON memiliki nilai true. Jika variabel SOUND_ON memiliki nilai true, maka program akan mengeluarkan suara kartu untuk menandakan perpindahan kartu. Kemudian program akan melakukan inisialisasi variabel ketemu sama dengan true (berarti kartu berhasil dipindahkan) dan statement break; untuk menghentikan perulangan for.

Kondisi jika menggambarkan penghapusan kartu yang sudah dipindah ke foundation tetapi kartu tersebut masih ada di class DealtCardsStack. Pertama-tama, program akan melakukan pengecekan nilai variabel ketemu dahulu apakah nilai tersebut true atau false. Jika nilai variabel ketemu adalah true, maka terjadi proses perpindahan kartu sebelumnya.

Jika nilai variabel ketemu adalah false, maka tidak terjadi proses perpindahan kartu sebelumnya. Jika nilai variabel ketemu adalah true, maka langkah selanjutnya program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class DealtCardsStack, maka penghapusan kartu akan dieksekusi dengan cara menampung nilai variabel CHOOSEN_CARD ke dalam variabel cards yang sudah di deklarasikan dengan tipe data Collection ArrayList dan selanjutnya variabel stack yang saat ini di DealtCardsStack akan mencari dan menghapus nilai data dari variabel cards tersebut dengan statement stack.removeCards(cards); Selanjutnya function repaint();akan dieksekusi agar program melakukan penggambaran ulang dari kartu yang dipindah dan dihapus.

Pengecekan foundation yang dimulai dengan inisialisasi nilai variabel fullCount sama dengan 0. Kemudian program akan melakukan perulangan for menggunakan variabel stack dengan tipe data class Stack dan ukuran datanya berdasarkan variabel stacks. Didalam perulangan for ini, program akan melakukan pengecekan jika variabel stack mengandung (instanceof) tipe data class foundation maka akan dilakukan pengecekan apakah nilai dari variabel stack tersebut penuh atau tidak dengan function isFull(). Jika function isFull() menghasilkan nilai true maka nilai variabel fullCount akan dinaikkan sebanyak 1 dan perulangan akan berlanjut sampai akhir.

Setelah perulangan for dengan variabel stack berakhir, maka langkah selanjutnya adalah pengecekan apakah nilai dari variabel fullCount sama dengan 4. Jika nilai dari variabel fullCount sama dengan 4 berarti masing-masing kartu di foundation sudah penuh dan program berakhir. Selanjutnya program akan mematikan waktu permainan dengan statement timerComponent.gameOver(), inisialisasi variabel GAME_OVER dengan nilai sama dengan true dan nilai variabel highScore dengan parameter timerComponent.getTime();

## IV. SIMPULAN

Kesimpulan dari penelitian ini adalah Algoritma Greedy dapat di implementasikan ke dalam permainan solitaitre, karena algoritma greedy akan mencari jalur dari kartu yang akan dipindahkan, karena pada saat permainan di mulai, algoritma greedy menyiapkan jalur-jalur keluarnya kartu secara tepat agar permainan dapat diselesaikan.

## V. DAFTAR RUJUKAN


[1]. S. Bhuvaneswari, et al. Man-Machine Interface, International Journal of Artificial Intelligence and Applications, vol. 3., 2012, pp 139-147.

[2]. W. Budiharto, & Derwin Suhartono. Artificial Intelligence, Andi, Yogyakarta. 2014.

[3]. K. Warwick. Artificial Intelligence: The Basics, Routledge, New York. 2012.

[4]. B, Youssef. Expert PC Troubleshooter with Fuzzy Logic and Self Learning Support, International Journal of Artificial Intelligence and Applications, vol. 3, 2012. pp 11-21.

[5]. I. Idris. Model and Algorithm in Artificial Immune System for Spam Detection, International Journal of Artificial Intelligence and Applications, vol. 3., 2012, pp 83-94.

[6]. S. Rusell, P. Norvig. Artificial Intelligence: A modern approach. Pearson, ISBN: 978-1-2920-2420-2. 2013.

[7]. D. Tegarden, et all. Systems Analysis and Design with UML/ 4th ed, John Wiley & Sons. 2013.