BAB III

ANALISIS DAN PERANCANGAN ALGORITMA DAN PERMAINAN OTHELLO

3.1 Analisis Algoritma

Salah satu dari tujuan penelitian dilakukan adalah untuk membandingkan manakah diatara algoritma *greedy* dan algoritma *minimax* yang lebih baik ketika menjadi lawan manusia dalam permainan *othello.* Maka dari itu di dalam analisis algoritma akan dibahas mengenai algoritma *greedy* dan algoritma *minimax* yang digunakan sebagai lawan dalam permainan *othello.*

3.1.1 Algoritma Greedy

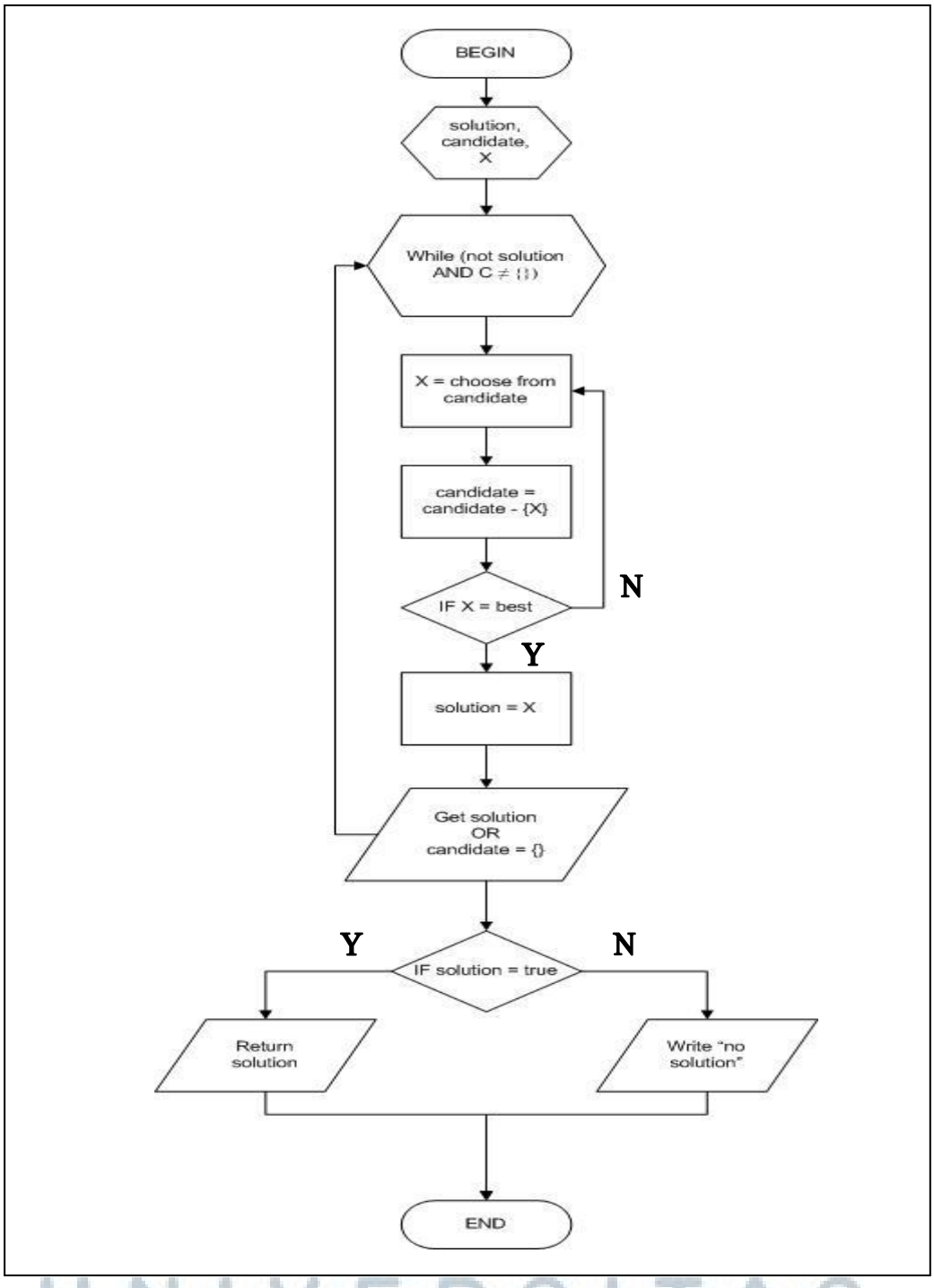
Algoritma *greedy* merupakan algoritma yang mencari solusi optimum lokal untuk menghasilkan solusi optimum global. Pada beberapa kasus, solusi yang ditawarkan oleh algoritma *greedy* mampu memberikan hasil yang optimal. Namun pada kasus lain, solusi yang ditawarkan oleh algoritma *greedy* tidak memberikan hasil yang memuaskan.

Pada kasus perancangan algoritma *greedy* sebagai lawan permainan dalam permainan *othello,* algoritma *greedy* yang diterapkan merupakan algoritma *greedy* yang berusaha menempati posisi dimana posisi tersebut dapat membalik bidak milik lawan paling banyak. Sesuai dengan prinsip dari algoritma *greedy* itu sendiri yang merupakan algoritma ‘rakus’, maka posisi yang dapat membalik jumlah bidak milik lawan paling banyak adalah solusi optimum lokal pada saat

permainan. Dengan solusi optimum lokal pada saat itu, tanpa mempertimbangkan solusi lain, maka diharapkan solusi optimum lokal saat itu adalah solusi optimum global sampai dengan akhir permainan.

Berdasarkan prinsip dari permainan *othello* dimana pemenang dalam permainan adalah yang memiliki bidak paling banyak, maka hipotesa awal yang dapat diambil dari penerapan algoritma *greedy* pada permainan *othello* adalah bahwa algoritma *greedy* dapat menjadi lawan yang handal dalam permainan *othello.*

Adapun prinsip kerja *artificial intelligent* dengan algoritma *greedy* dapat dilihat pada *flowchart* gambar 3.1.



Gambar 3.1 *Flowchart Artificial Intelligent* Algoritma *Greedy*

Pada *flowchart*, prinsip kerja algoritma *greedy* diawali dengan pengecekan posisi yang memungkinkan bagi pemain untuk menempatkan bidak. Kemudian dilakukan penghitungan berapa banyak jumlah bidak musuh yang dapat dibalik oleh pemain jika bidak ditempatkan pada posisi tersebut. Pengecekan dilakukan di tiap posisi yang memungkinkan untuk diletakkan bidak. Kemudian dilakukan perbandingan jumlah bidak musuh yang dapat dibalik di masing-masing posisi yang memungkinkan tersebut. Posisi yang dapat membalik bidak musuh paling banyak itulah posisi yang dipilih oleh *artificial intelligent* dengan algoritma *greedy*.

Dari *flowchart* yang terdapat pada gambar 3.1 diimplementasikan kedalam program seperti pada potongan *source code* 3.1.

Source Code 3.1 Algoritma Greedy

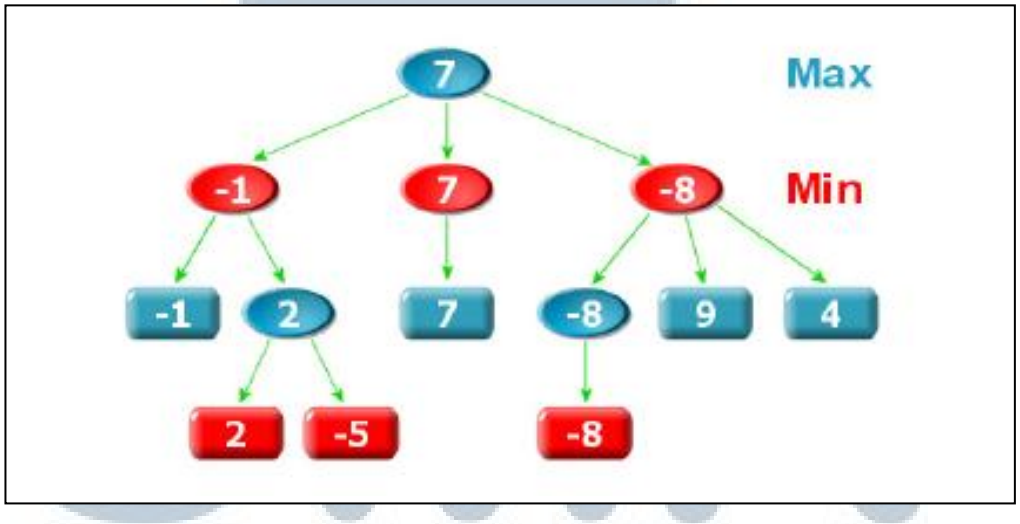
```
int max = 0;
int bestX = 0, bestY = 0;
for (int i = 0; i < 8; i++) {
      for (int j = 0; j < 8; j++) {
            int cnt = ValidCell(i, j, musuh(turn), false);
            if (cnt > max) {
                  max = cnt;
                  bestX = i;
                  bestY = j;
            }
      }
}

ValidCell(bestX, bestY, musuh(turn), true);
if (turn == 1) {
      turn = 2;
} else if (turn == 2) {
      turn = 1;
}
boardgame[bestX][bestY] = turn;
```

3.1.2 Algoritma Minimax

Berbeda dengan algoritma *greedy,* algoritma *minimax* memiliki konsep yang melihat beberapa langkah ke depan. Algoritma *minimax* berupaya membuat solusi dengan cara membuat pohon permainan yang memiliki suatu kedalaman tertentu.

Algoritma *minimax* yang melihat beberapa langkah kedepan berasumsi bahwa setiap langkah yang diambil oleh musuh adalah langkah terbaik saat itu. Kemudian dibuat kembali langkah selanjutnya sampai suatu kedalaman tertentu.

Prinsip algoritma *minimax* dapat dilihat pada gambar 3.2. dimana semakin tinggi nilai suatu cabang, maka semakin bagus bila bidak diletakkan pada posisi tersebut.

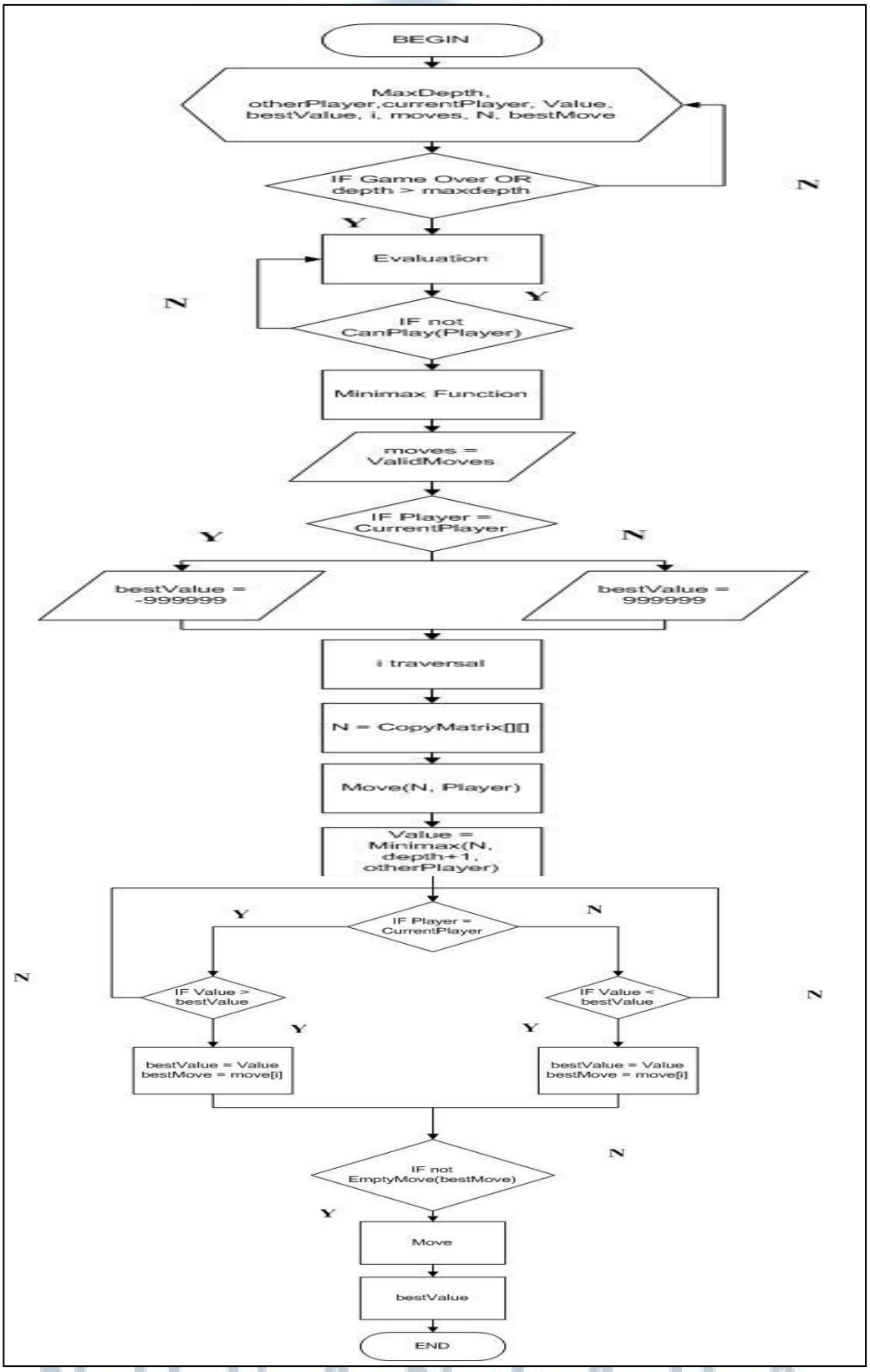


Gambar 3.2 Prinsip Algoritma *Minimax*

Berdasarkan prinsip algoritma *minimax* yang melihat beberapa langkah kedepan dan memprediksi langkah yang diambil musuh adalah langkah yang paling baik walaupun saat musuh memperoleh giliran belum tentu menempatkan bidak di posisi yang terbaik sesuai dengan prediksi, maka hipotesa awal yang

dapat diambil dari penerapan algoritma *minimax* adalah bahwa algoritma *minimax* merupakan lawan yang handal dalam permainan *othello* karena mampu memperkirakan langkah yang diambil oleh musuh.

Penerapan algoritma *minimax* sebagai *artificial intelligent* dapat dilihat pada *flowchart* gambar 3.3. Pada gambar dapat dilihat bahwa pertama-tama diawali dengan menyimpan terlebih dahulu posisi papan permainan pada awal permainan. Kemudian dilakukan pengecekan sampai dengan kedalaman tertentu. Pengecekan yang dilakukan adalah dengan mensimulasikan permainan di setiap posisi yang memungkinkan untuk diletakkan bidak permainan. Dari peletakkan bidak di setiap posisi sampai dengan kedalaman tertentu, musuh diprediksi mengambil langkah terbaik pada setiap pergantian giliran. Hasil akhir yang akan diperoleh berupa nilai-nilai di setiap kemungkinan posisi untuk menempatkan bidak. Posisi yang memiliki nilai paling tinggi merupakan posisi yang akan dipilih oleh *artificial intelligent* dengan algoritma *minimax.*



Gambar 3.3 *Flowchart Artificial Intelligent* Algoritma *Minimax*

Dari *flowchart* yang terdapat pada gambar 3.3 diimplementasikan kedalam program seperti pada potongan *source code* 3.2.

Source Code 3.2 Algoritma Minimax

```
int i,j;
int maxdepth = 8;
int bnyk = 8 * 8;
int[] arr = new int[bnyk];
int[] temparr = new int[bnyk];

if (depth >= maxdepth){
     return best;
}

int jum = -9999 * idx;
int value = 0;
for (int ctr = 0; ctr < bnyk; ++ctr){
     temparr[ctr] = arr[ctr] = arr2[ctr];
}

int count = 0;
for (int ctr = 0; ctr < bnyk; ++ctr){
     j = arraycek[ctr] % 8;
     i = arraycek[ctr] / 8;
     if (arr[arraycek[ctr]] == 0 && cekCPU(i, j,
     arr,status)){
          count++;
          arr[(i * 8) + j] = status;
          if (depth < maxdepth){
               value = 0;
               value += arraycektmp[arraycek[ctr]];
               for (int ctr2 = 0; ctr2 < bnyk; ++ctr2)
                    if (arr[ctr2] == status)
                         value++;
                    value *= idx;
                    value += nilai;
                    int tmp = AI(depth + 1, arr,
                    status, value, jum, -idx,
                    count);
                    if (depth % 2 == 0)
                         if (tmp > jum)
                              jum = tmp;
                    else
                         if (tmp < jum)
                              jum = tmp;
```

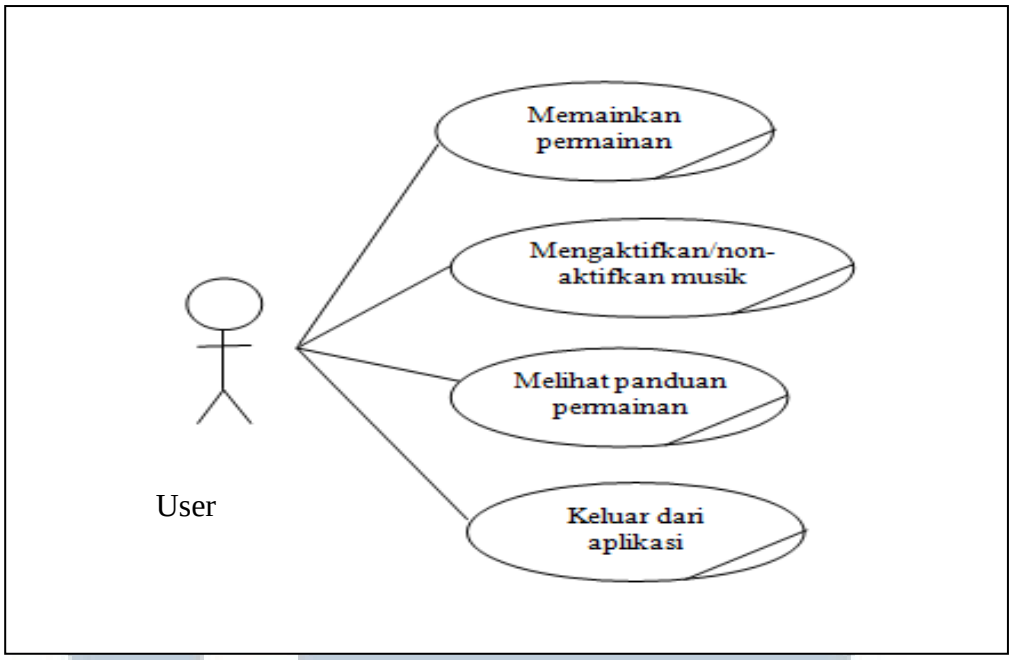
```
                    if (depth % 2 == 0)
                        if (jum > minmax)
                            return minmax;
                    else
                        if (jum < minmax)
                            return minmax;
                }
            }
        }
```

3.2 Perancangan Sistem

Pada perancangan sistem akan digambarkan proses perancangan sistem mulai dari *use case, class diagram, sequence diagram, activity diagram, deployment diagram,* hingga perancangan antarmuka. Masing-masing perancangan diagram tersebut mewakili penjelasan aktivitas, arus interaksi, dan proses yang nantinya berjalan pada sistem.

3.2.1 Use Case

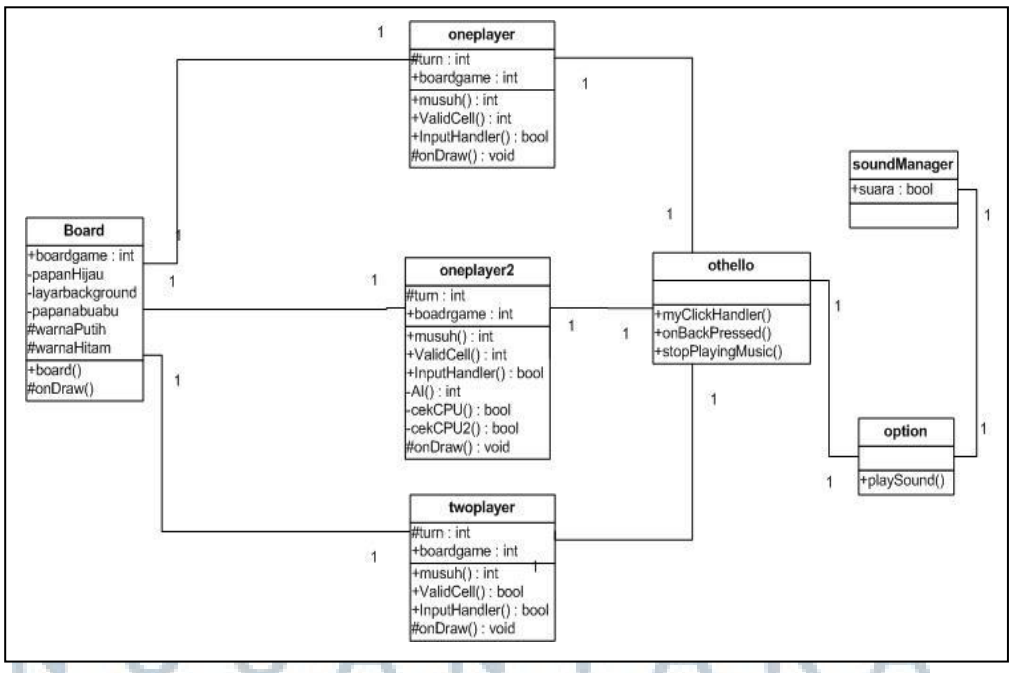
Use Case dari sistem yang dirancang dapat dilihat pada gambar 3.4, dimana pada gambar dapat dilihat aktivitas apa yang dapat dilakukan oleh pemain.



Gambar 3.4 *Use Case*

3.2.2 Class Diagram

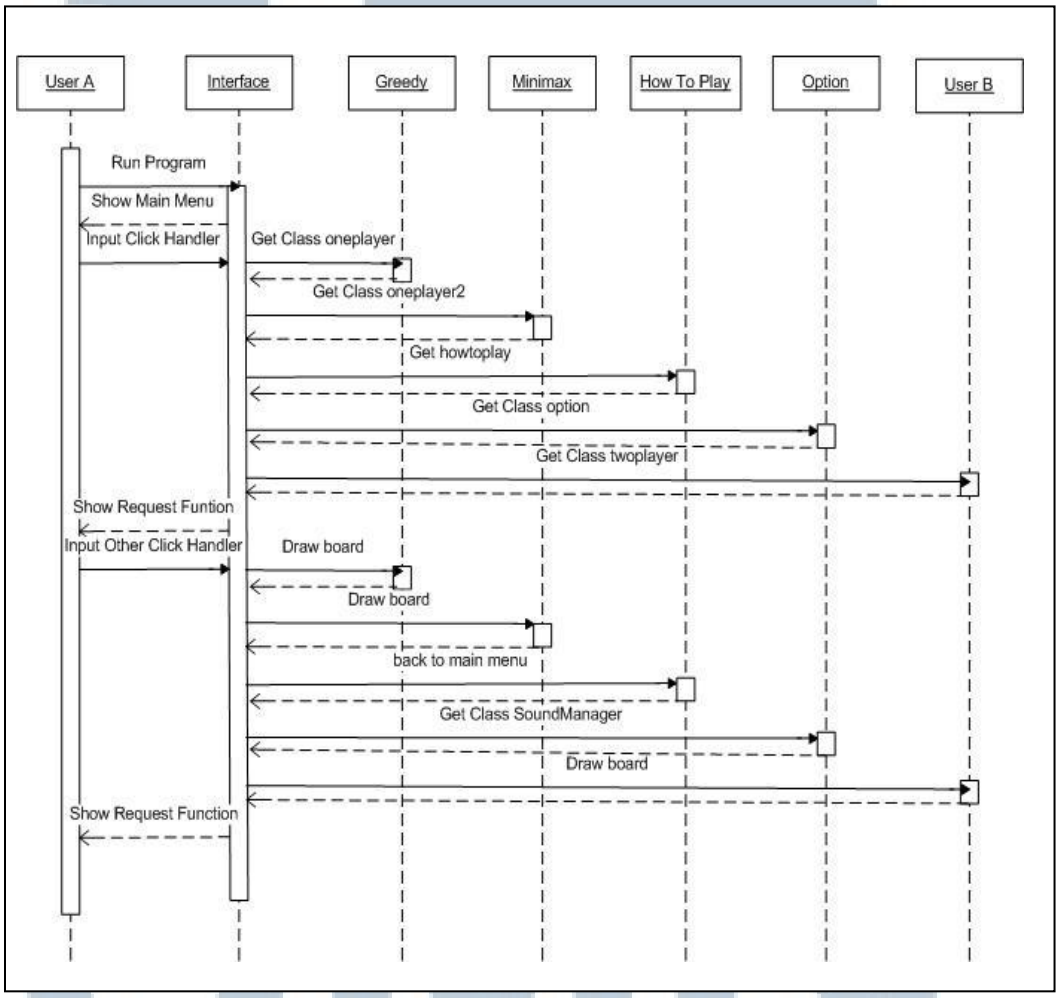
Class diagram akan menjelaskan struktur *class* yang ada di dalam sistem. Adapun struktur *class* tersebut dapat dilihat pada gambar 3.5.



Gambar 3.5 *Class Diagram*

3.2.3 Sequence Diagram

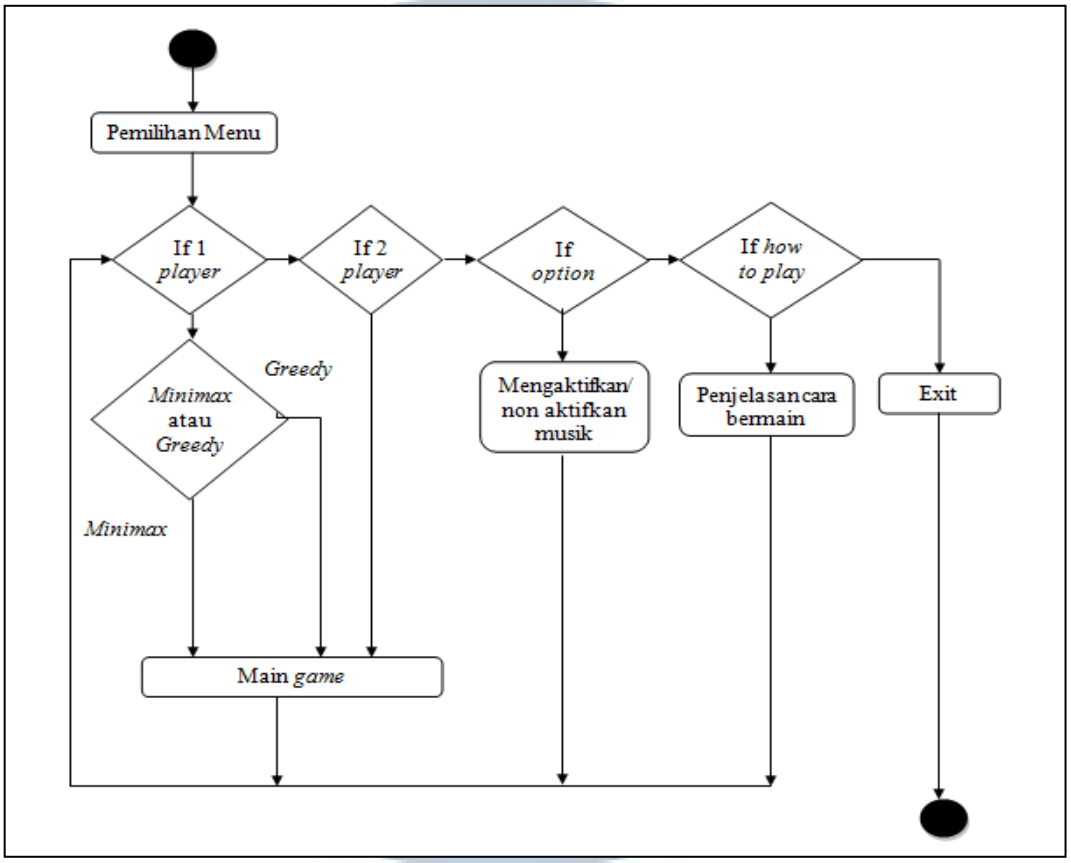
Sequence diagram menggambarkan alur interaksi proses yang terjadi di dalam sistem. Proses interaksi dapat dilihat pada gambar 3.6.



Gambar 3.6 *Sequence Diagram*

3.2.4 Activity Diagram

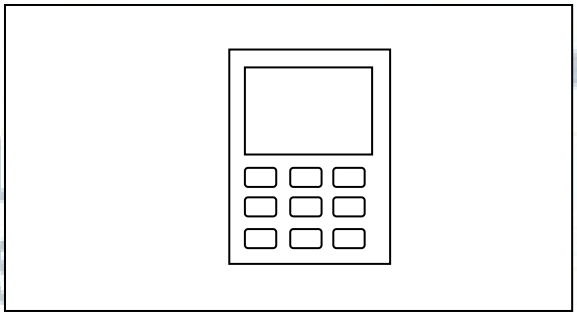
Activity diagram menggambarkan keseluruhan proses kerja yang terjadi di dalam sistem. Keseluruhan proses kerja yang terjadi dari awal hingga akhir dapat dilihat pada gambar 3.7.



Gambar 3.7 *Activity Diagram*

3.2.5 Deployment Diagram

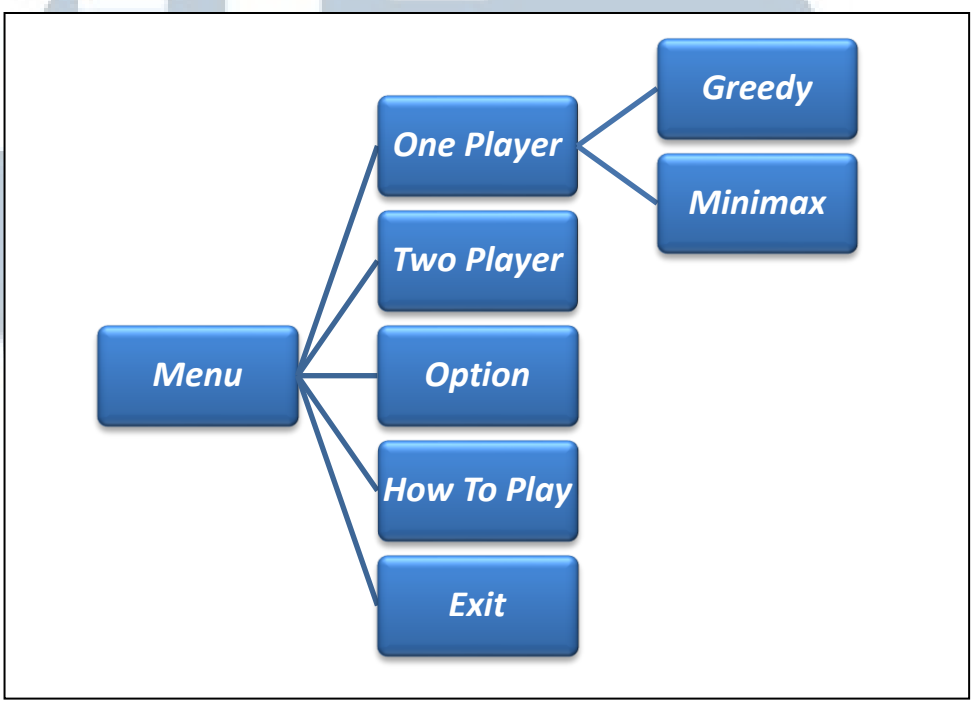
Deployment diagram merupakan perangkat fisik yang berhubungan dengan sistem. Dalam hal ini perangkat fisik yang digunakan hanyalah sebuah ponsel, maka dari itu *deployment diagram* yang terdapat pada gambar 3.8 hanyalah sebuah ponsel.



Gambar 3.8 *Deployment Diagram*

3.2.6 Perancangan Antarmuka

Struktur navigasi menu yang terdapat dalam aplikasi permainan *othello* yang dibuat dapat dilihat pada gambar 3.8.



Gambar 3.9 *Top Down Design Menu*

Menu utama aplikasi dibagi menjadi 5 bagian, kemudian pada menu "*one player*" pemain kemudian akan memilih lawan terlebih dahulu, apakah lawan dengan *artificial intelligent* yang berpikir dengan menggunakan algoritma *greedy* atau *artificial intelligent* yang berpikir dengan menggunakan algoritma *minimax.*

Pada menu "*two player"* pemain akan langsung memulai permainan namun tidak melawan *artificial intelligent* melainkan melawan sesama manusia dimana proses permainan dilakukan secara bergantian pada satu *device*.

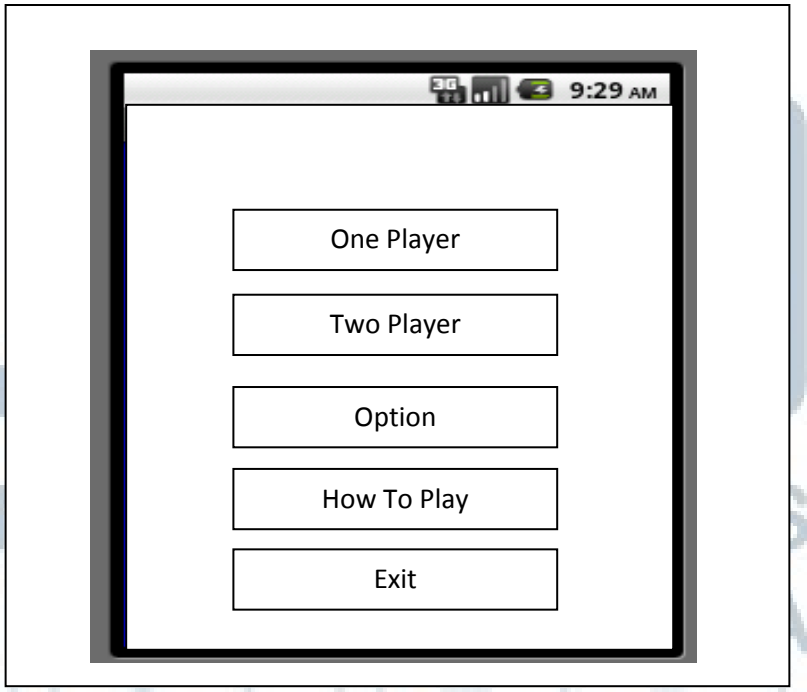
Pada menu "*option*" pemain dapat meng-aktifkan atau me-non-aktifkan musik pengiring permainan. Dan pada menu "*how to play*" pemain dapat

membaca teks berisi narasi penjelasan bagaimana cara bermain dalam permainan *othello*.

Menu exit merupakan menu untuk keluar dari aplikasi. Setelah menu ini dipilih, pengguna ponsel berbasis android tidak langsung keluar dari aplikasi melainkan melakukan konfirmasi terlebih dahulu apakah pengguna ponsel yakin akan keluar dari aplikasi permainan atau tidak.

3.2.7 Desain Antarmuka

Desain tampilan antarmuka (*interface*) yang akan diimplementasikan ke dalam sistem memiliki 4 buah *file* XML, yang terdiri atas *main*.xml, *opponent*.xml, *option*.xml, dan *howtoplay*.xml. Sedangkan untuk menggambar papan permainan menggunakan fungsi *paint.*



Gambar 3.10 Desain Tampilan Antarmuka Halaman Utama

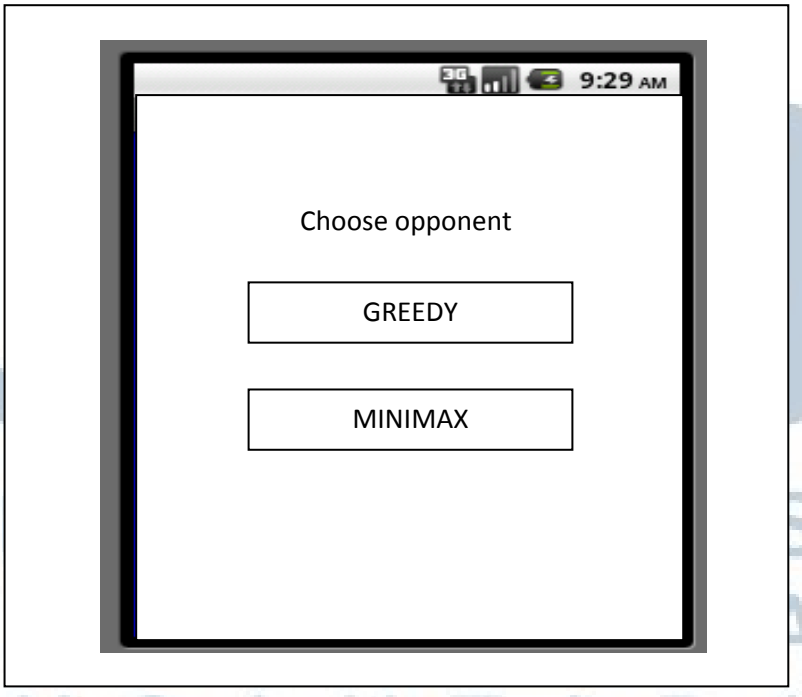
Pada gambar 3.10 dapat dilihat desain tampilan untuk halaman utama ketika aplikasi dijalankan. Desain tersebut dirancang dengan menggunakan *file* XML dengan nama *main*.xml. Pada *source code* 3.3 dapat dilihat isi dari *file main*.xml yang digunakan untuk membangun menu utama pada gambar 3.10.

Source Code 3.3 XML Halaman Utama

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical" android:padding="40dip"
      android:layout_width="fill_parent"
android:layout_height="fill_parent"
      android:background="@color/blue"
android:screenOrientation="portrait">
      <TextView android:textSize="25sp"
android:layout_gravity="center_horizontal"
            android:text="IPOTHELLO GAME"
android:layout_height="wrap_content"
            android:textColor="@color/black"
android:textStyle="bold"
            android:layout_width="wrap_content"
android:id="@+id/textView1"></TextView>
      <Button android:id="@+id/singleplayer"
android:text="Single Player"
            android:layout_width="fill_parent"
android:layout_height="wrap_content"
            android:layout_gravity="center"
android:onClick="myClickHandler"></Button>
      <Button android:id="@+id/twoplayer" android:text="Two
Player"
            android:layout_width="fill_parent"
android:layout_height="wrap_content"
            android:layout_gravity="center"
android:onClick="myClickHandler"></Button>
      <Button android:id="@+id/option" android:text="Option"
            android:layout_width="fill_parent"
android:layout_height="wrap_content"
            android:layout_gravity="center"
android:onClick="myClickHandler"></Button>
      <Button android:id="@+id/howtoplay" android:text="How
To Play"
            android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

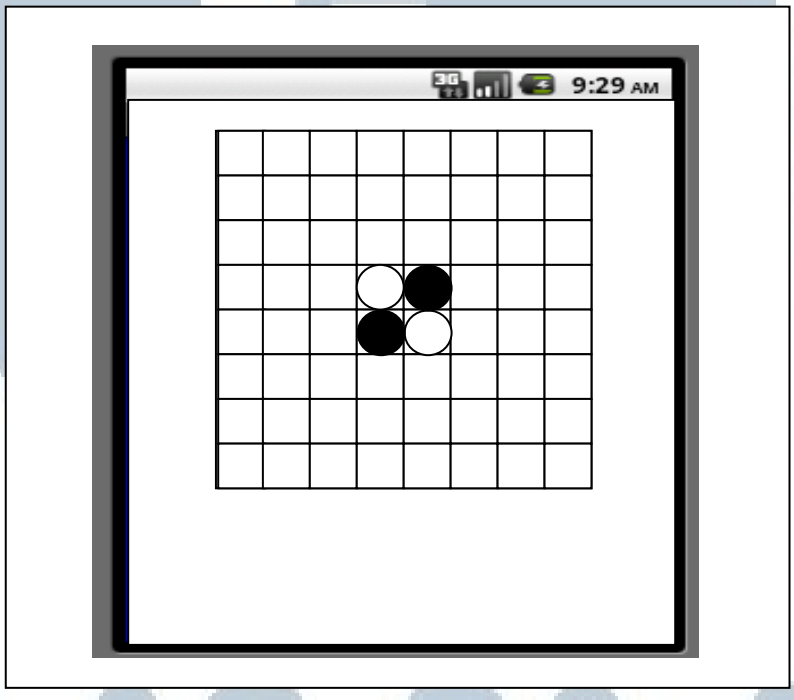
```
            android:layout_gravity="center"
android:onClick="myClickHandler"></Button>
      <Button android:id="@+id/exit" android:text="Exit"
            android:layout_gravity="center"
android:layout_width="fill_parent"
            android:layout_height="wrap_content"
android:onClick="myClickHandler"></Button>
      <TextView android:layout_gravity="center_horizontal"
            android:text="©Ivan Prakasa"
android:layout_height="wrap_content"
            android:textColor="@color/black"
android:textStyle="bold"
            android:layout_width="wrap_content"
android:id="@+id/textView1"></TextView>
</LinearLayout>
```

Pada gambar 3.11 dapat dilihat tampilan pemilihan lawan permainan. Tampilan ini muncul ketika pemain memilih menu "*one player*". Untuk membangun tampilan seperti pada gambar 3.11 pada program digunakan *file* dengan nama *opponent*.xml.



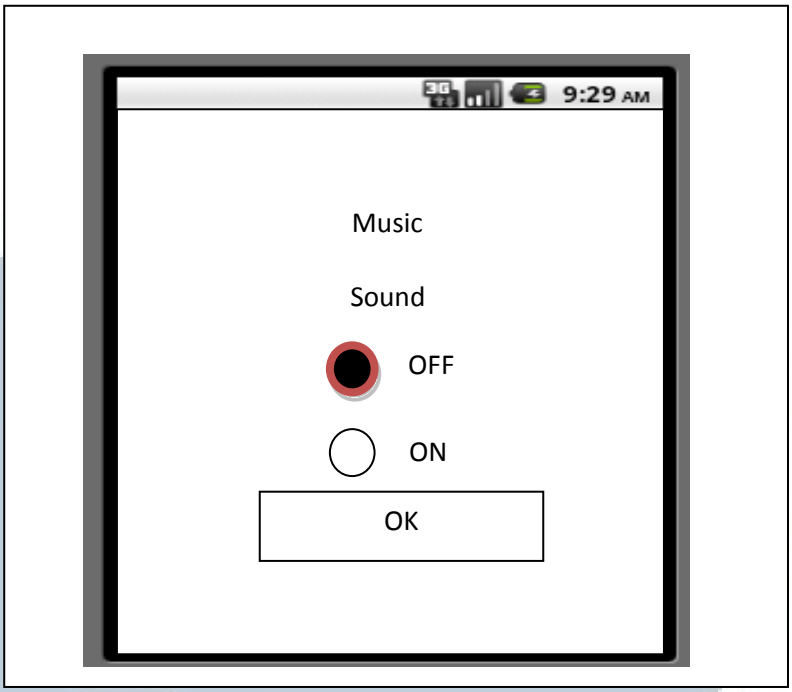
Gambar 3.11 Desain Tampilan Pemilihan Lawan Permainan

Pada gambar 3.12 dapat dilihat rancangan tampilan untuk papan permainan. Tampilan ini dirancang menggunakan fungsi *paint* yang digunakan untuk menggambar pada *canvas*. Untuk merancang tampilan pada gambar 3.12 digunakan *file* dengan nama *board.java*.

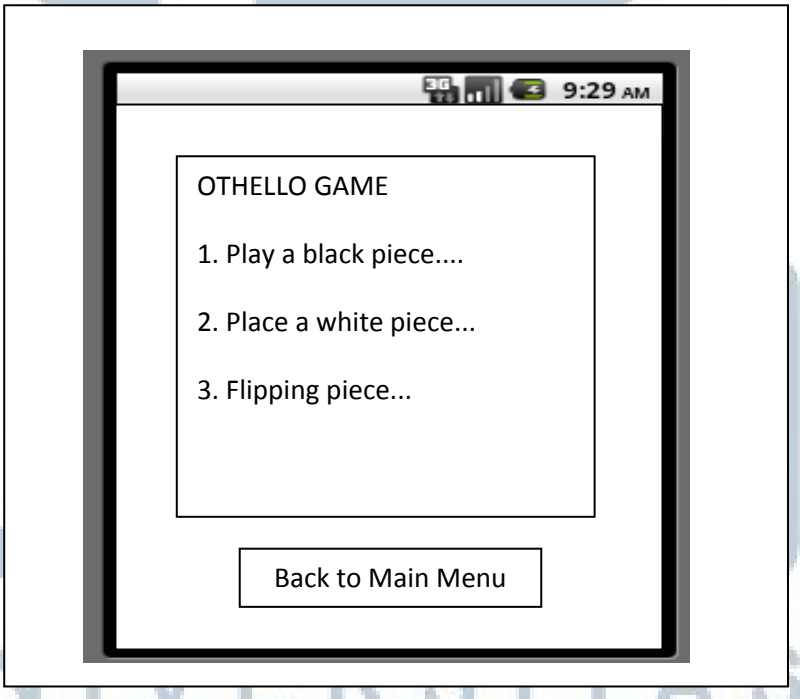


Gambar 3.12 Desain Tampilan Papan Permainan

Pada gambar 3.13 dan gambar 3.14 dapat dilihat rancangan tampilan layar untuk menu "*option"* dan *"how to play"* yang implementasinya menggunakan *file* XML seperti pada rancangan gambar 3.11 dan 3.10, yang pada program diberi nama *option*.xml dan *howtoplay*.xml.



Gambar 3.13 Desain Menu "*Option*"



Gambar 3.14 Desain Menu "*How To Play*"