4. IMPLEMENTASI SISTEM

Pada bab ini akan dijelaskan tentang implementasi dari sistem yang dibuat pada tugas akhir ini. Secara umum program dibagi atas dua bagian. Yang pertama ialah program pada sisi *server* yang terdiri dari beberapa program PHP. Dan yang kedua ialah program aplikasi utama yaitu program *game multiplayer* yang diinstall pada sisi *handphone user* atau *client* menggunakan J2ME.

4.1. Implementasi Sistem pada Sisi *Server*

Untuk implementasi sistem di sisi *server*, perlu melakukan instalasi *software-software* pendukung agar bisa menjadi Web *Server*. *Software* yang harus diinstal antara lain program Apache Web *Server*, PHP dan MySQL dimana proses instalasinya relatif mudah karena tidak perlu melakukan konfigurasi yang rumit.

Yang patut diperhatikan ketika instalasi yaitu saat melakukan *editing file* 'httpd.conf' dimana perlu melakukan beberapa penambahan perintah berkaitan dengan penggunaan *module* PHP untuk penanganan *script* PHP oleh *server* Apache. Cara instalasi dan konfigurasi Apache, PHP dan MySQL secara singkat, telah dilampirkan pada bagian Lampiran 2 sampai 4.

Khusus untuk pembuatan *gameserver* ini, diharapkan *server* yang dibangun mampu menangani hingga 400 *user* yang *online* secara bersamaan (99 *room* @ 4 *user*). Karena secara *default*, *user* yang mampu ditangani berjumlah 150 *user* saja, maka pada file 'httpd.conf' perlu dilakukan penambahan nilai. Menggunakan program *editor* (notepad, wordpad, dll.) buka file 'httpd.conf' yang ada pada direktori conf pada *folder* Apache (contoh: c:\apache\conf\httpd.conf). Masukkan nilai '`MaxClients 400`' pada file tersebut. Contohnya sebagai berikut:

```
...

MaxClients 400

# 400 adalah jumlah client maksimum yang mampu ditangani
# server ini secara bersamaan
...
```

S*erver* menggunakan PHP sebagai bahasa *script* untuk melakukan pengambilan data dari *database* MySQL. Program PHP pada desain sistem ini

terdiri atas 11 *file*. Berikut akan dijelaskan satu per satu file PHP yang digunakan dalam pembuatan sistem di server.

4.1.1. 'setting.php'

File 'setting.php' digunakan untuk membuat koneksi ke *database* MySQL. Ketika akan melakukan konesi ke *database*, setiap akan melakukan koneksi perlu melakukan insialisasi koneksi yang meliputi alamat komputer (IP atau *Internet Protocol*) yang digunakan, nama dan *password* administrator *database* menggunakan perintah `mysql_connect`. Setelah itu dilakukan pemilihan nama *database* yang pada tugas akhir ini diberi nama '*gameserver*' menggunakan perintah `mysql_select_db`.

Semua *file* PHP yang digunakan pada sistem ini, melaksanakan pengolahan ke *database* sehingga dengan tujuan efesiensi, inisialisasi dituliskan ke satu *file* saja yaitu 'setting.php'. Pada penggunaannya, *file* ini cukup disertakan atau di-*include*-kan ke semua *file* yang melakukan koneksi ke *database*. Isi dari *file* ini sebagai berikut :

```
$conn = mysql_connect("202.43.253.54","root","adminroot")
      or die ("koneksi database error");
mysql_select_db("gameserver",$conn);
```

4.1.2. '*login*.php'

File '*login*.php' digunakan ketika proses *authentifikasi user* sebelum masuk ke menu utama permainan. Ketika *handphone* mengakses *file* ini, variabel nama dan *password* yang telah *user* masukkan di form akan ikut disertakan. Nama dan *password* tersebut selanjutnya dicocokkan dengan record nama dan *password* yang sudah ada di tabel *login*.

Yang dilakukan oleh program '*login*.php' pertama kali adalah pengambilan data nama *user* dari tabel *login*.

```
$log_SQL="select * from login where name = '$name';
$log_sult= mysql_query($log_SQL);
$log_row =  mySQL_fetch_array($log_sult);
```

Bila nama *user* terdapat di dalam tabel *login*, selanjutnya dilakukan pengecekkan nilai *password* yang dimasukkan *user*. Bila *password* yang dicocokkan benar, maka dilakukan pengecekan ke tabel *useronline* untuk memeriksa apakah ada *user*

dengan nama yang sama yang *online*. Hal ini mencegah adanya account *user* yang digunakan lebih dari satu *user* dalam satu waktu.

```
...
if ($log_row>0) {
if($password == $log_row[password]) {
$sess_cek="select * from useronline where name = '$name'";
$stat_cek= mysql_query($sess_cek);
$stat_sult = mySQL_fetch_array($stat_cek);
   if ($stat_sult>0){
print("Your status is online from different
machine. Please logout first");
exit();
...
```

Bila nama *user* tidak didapatkan dari tabel *useronline* (dengan kata lain *user* belum *online*) maka nama *user* dimasukkan ke field name pada tabel *useronline.* Selanjutnya dilakukan pengiriman nilai balik (respon) ke *handphone* sebagai konfirmasi bahwa nama dan *password* yang dimasukkan *user* cocok dengan yang ada di *database*. Karena nama *user* tercantum pada tabel *useronline*, maka *user* dianggap *online* atau aktif.

```
{ $into_session = "insert into useronline (name)
   values ('$name')";
$sess_sult = mySQL_query($into_session,$conn);
if ($sess_sult) { print ("user100%password"); } } }
```

4.1.3. ‘see*room*.php’

File ‘see*room*.php’ digunakan untuk memberikan informasi dari *database* ke *user* tentang *room* yang (masih) ada beserta statusnya serta *user* yang berada di dalam *room-room* yang masih buka. Pengecekan meliputi *user* yang *online*, kondisi *room* (status permainan), jumlah *room* dan nama pemain di dalam *room* yang (masih) ada. Selain itu, respon dari *file* ini berupa nilai *room* yang bisa dimasukkan bila ingin membuat *room* baru.

Proses dimulai dengan pencarian nama *user* lain yang sedang *online* saat itu, melalui pengecekan nama dari tabel *useronline*. Bila hasil pengecekan menunjukkan tidak terdapat *user* lain, maka respon ‘no*user*’ akan dikirmkan ke *handphone*.

```
$strSQL="select name from useronline where name != '$name'";
$result = mysql_query($strSQL);
$i=1; $n=1;
while ($row = mySQL_fetch_row($result))
   { $res[$i] = $row[0]; $i++; }
$i--;
// bila tidak ada user lain
if($i<=0) { print("nouser"); }
```

Bila terdapat *user* lain yang sedang *online*, maka pengecekan dilanjutkan dengan pengecekan *room* yang ada. Hasil pengecekan *room* selanjutnya dikirimkan ke *handphone*.

```
...
   else {$roomSQL="select room from useronline
      order by room desc " ;
   $res0 = mysql_query($roomSQL);
   $ro = mySQL_fetch_row($res0);
   if ($ro[0]=='0')
      {print("Belum ada room yang dibuat\n");
      exit();}
   else
      {print("Terdapat ".$ro[0]." room.\n");
...
```

Setelah jumlah *room* didapat, dilakukan pengecekan pada setiap *room* untuk melihat kondisi *room*, apakah *user* masih bias *join* ke *room* atau sudah ditutup/*closed*.

```
...
   for($a=1;$a<=$ro[0];$a++){
      print(" Room ".$a." : ");
      $i=0;
      $strSQL="select name from useronline where
         room = '$a' and play = '1'";
      $result = mysql_query($strSQL);
      while ($row = mySQL_fetch_row($result)) { $i++; }
...
```

Kondisi *closed* didapatkan bila pada suatu *room* memiliki record play yang bernilai 1, dimana dianggap permainan sedang berjalan di *room* tersebut. Bila status playnya tidak sama dengan 1, maka dilakukan pengambilan data nama *user* yang berada di dalam *room* demikian proses ini diulang terus hingga pengecekan pada *room* terakhir yang sudah dibuat.

```
...
      {print ("closed.\n"); }
   else{
      $rSQL="select name from useronline where room ='$a'";
         $result = mysql_query($rSQL);
         $i=0; $n=1;
         while ($row = mySQL_fetch_row($result))
            {$i++; $res[$i] = $row[0]; }
      while ($n<=$i)
         { print("  ".$res[$n]); $n++; }
         print (".\n"); } }

      // tampilkan pesan untuk membuat room baru
      print ("Masukkan '".(($ro[0]+1)). "' bila ingin
         membuat room");
...
```

4.1.4. '*joinroom*.php'

File '*joinroom*.php' digunakan saat *client* atau *user* memasukkan suatu nilai (angka) ke dalam form *join room* untuk bisa masuk ke *room* yang diminta. Aplikasi *handphone* akan mengirimkan *request* ke *server* berupa variabel name dan nilai *room* untuk pemrosesan *join room*.

Menggunakan nilai *room* yang dimasukkan *user* pada *handphone*, dilakukan pengambilan data ke *database* yang memiliki nilai *room* yang sama dengan yang diminta *user*. Data yang juga berisi nama-nama *user* di dalam *room*, selanjutnya dicocokkan dengan nama *user*. Hal ini dilakukan untuk mengecek apakah *user* telah *join* dengan *room* itu sebelumnya.

```
$strSQL = "select * from useronline where room='$newroom'";
$result = mysql_query($strSQL);
$i=0;
while($row = mySQL_fetch_row($result))
   { $i++;
   $res[$i] = $row[0];
   $roomnya[$i] = $row[2];
if($res[$i] == $name)
   { $x = youhere; }    }
if($x == youhere)
   { print("youalready"); }
```

Bila *user* setelah dicek belum tergabung dengan *room* yang diminta, maka pengecekan dilanjutkan dengan pengambilan status permainan yang ada pada *room*. Bila dinyatakan penuh maka konfirmasi bahwa *user* tidak bisa *join* ke *room* dikirimkan ke *handphone*.

```
{ if ($roomnya[1] == 1)
   { print("roomfull"); }
else if  ($i==4)
   { print("roomfull"); }
```

Permainan ini menggunakan batasan pemain di setiap *room*nya, dimana bila terdapat 4 orang di *room*, maka permainan langsung diaktifkan. Bila telah terdapat 3 orang di *room*, maka orang ke-4 yang *join* dengan *room* akan melakukan pengaktifan status permainan dan selanjutnya permainan di *room* tersebut akan dimulai.

```
$playte="update useronline set play = '1',
      score = '-1' where room = '$newroom'";
$upd_result = mysql_query($playte);
print ("playnow"); }
```

Bila semua kondisi diatas tidak ada yang terpenuhi (nilai *room* yang dimasukkan *user* sesuai, *room* belum penuh, status permainan belum aktif), maka proses yang dilakukan adalah memasukkan nilai *room* ke dsatabase *user* sehingga *user* dianggap berhasil *join* dengan *room* yang diminta.

```
{ $roomte="update useronline set room = '$newroom',
      play = '0', score = '-1' where name = '$name'";
$upd_result = mysql_query($roomte);
print("success"); } }
```

4.1.5. ‘numplyr.php’

File ‘numplyr.php’ digunakan saat setelah *user join* dengan suatu *room*. *User* yang baru *join* dengan *room* akan dimasukkan ke dalam *waiting room*. Pengecekan awal dimulai dengan pengecekan status permainan, apakah ada *user* lain yang ingin melakukan permainan tanpa menunggu *room* penuh. Bila ada yang mengaktifkan, kirimkan respon ke *handphone user* dan permainan dijalankan. Perintah yang digunakan untuk itu adalah :

```
$i=0;
$prSQL="select * from useronline where room = '$room'
   and play = '1'" ;
$res = mysql_query($prSQL);
while ($row = mySQL_fetch_row($res))
   { $i++;  }
//bila ada, kirim respon permainan mulai ke handphone
if ($i>0){ print ("playin"); }
```

Selanjutnya bila status permainan belum aktif maka pengecekan dilanjutkan dengan mengambil jumlah *user* yang ada di *room* dimana *user* berada sekarang. Proses diulang-ulang hingga tidak ditemukan data yang memiliki nilai *room* yang sama dengan *user*. Perintah yang digunakan yaitu :

```
{ $i=0;
$strSQL="select * from useronline where room = '$room' and
   name != '$name'" ;
$result = mysql_query($strSQL);
while ($row = mySQL_fetch_row($result))
   { $i++;  }
```

Bila proses pencarian selesai, akan dapatkan nilai jumlah *user* yang ada di *room* tersebut. Pengiriman respon ke *handphone* akan dilakukan bila ternyata tidak terdapat *user* lain di dalam *room*. Bila didapati *user* lain di dalam *room*, proses dilanjutkan dengan mengambil record nama *user* dan proses berulang sebanyak

jumlah *user* yang terdapat di dalam *room*. Hasilnya kemudian dikirimkan ke *handphone* sebagai respon.

```
if($i==0) { print ("nowait"); }
else {
   $i=0;
   print ("usname");
   $wiSQL="select * from useronline where room = '$room' ";
   $res = mysql_query($wiSQL);
while ($re = mySQL_fetch_row($res)) {
   $naplyr[$i] = $re[0];
   $scrplyr[$i] = $re[3];
   $i ++;}
print(($i)." user telah join room ".$room." :\n");
for ($a=0; $a<$i ;  $a++) {
   print ($naplyr[$a]." \n"); }
```

Saat di *waiting room*, *file* ‘numplyr.php’ akan diakses secara berkala dengan rentang waktu tertentu. Hasil respon akan menampilkan informasi jumlah *user* dan nama *user* di dalam *room* saat penegcekan dilakukan. Bila ditemukan *user* di dalam *room*, maka pilihan ‘PlayNow’ akan muncul di *handphone*. Pengecekan akan dihentikan bila hasil respon dari *server* memberikan informasi bahwa *room* sudah penuh atau ada *user* lain yang memulai permainan.

4.1.6. ‘freplay.php’

Bila *user* memilih opsi ini, maka *handphone* akan *request* ke *server* dengan memanggil *file* ‘freplay.php’.

Fungsi dari *file* ini adalah mengaktifkan status permainan pada semua *user* yang berada pada *room* yang sama dengan *user* yang me*request* nya, sehingga bila satu orang mengaktifkan, maka dapat dikatakan bahwa permainan di *room* tersebut telah diaktifkan. Kondisi tersebut bisa terpenuhi bila terdapat *user* lain di *room* tersebut yang sudah siap bermain.

```
$i = 0;
$strSQL="select * from useronline where room = '$room'
   and name != '$name' and score = '-1'" ;
$result = mysql_query($strSQL);
while ($row = mySQL_fetch_row($result))
   { $i++;  }

if($i<1)
   { print ("cannotplay"); }
else  {
   $roomte="update useronline set play = '1', score = '-1'
      where room = '$room'";
   $upd_result = mysql_query($roomte); }
```

4.1.7. ‘sendscore.php’

Setelah *user* melakukan 3 kali kesempatan melempar dadu saat bermain, nilai total yang didapat akan dikirimkan ke *server* untuk dilakukan proses perbandingan score dengan *user* lain.

Proses pertama yang dilakukan saat program menerima *request* dari *handphone* adalah mencocokkan variable nama dan *room user*. Cara yang digunakan adalah sebagai berikut :

```
$strSQL= "select * from useronline where
   name = '$name' AND room='$room'";
$result = mysql_query($strSQL,$conn);
$arry = mySQL_fetch_array($result);
```

Bila nama yang dicocokkan terdapat di tabel *useronline*, nilai yang disertakan (saat me-*request*) tersebut dimasukkan ke field score menurut *user* dan *room* yang dimaksud. Proses peng-*update*-an dilakukan dengan perintah seperti di bawah ini :

```
{$update="update useronline set score = '$score'
   where name = '$name' AND room='$room'";
$upd_result = mysql_query($update);
print("scorereceived"); }
```

4.1.8. ‘checkwin.php’

File ‘checkwin.php’ digunakan setelah permainan selesai dan pengiriman nilai ke *server* berhasil dilakukan, dimana *handphone* secara otomatis melakukan *request file* ini ke *server*. Adapun proses yang terjadi pada *file* ‘checkwin.php’ pada mulanya meliputi pengecekan keberadaan nilai dari *user* lain pada *room* tempat *user* bermain. Bila nilainya sama dengan -1, maka *user* dianggap belum memasukkan nilai ke *database* di *server*. Perintah yang digunakan untuk mengecek apakah ada *user* yang belum memasukkan nilai, sebagai berikut :

```
$winSQL="select name from useronline where
   room = '$room' and score = '-1'";
$rest = mysql_query($winSQL);
$i=0;
while (mySQL_fetch_row($rest))
   {$i++;}
```

Bila semua *user* telah memasukkan nilanya, selanjutnya dilakukan pengurutan data berdasarkan nilai record score pada *room* tersebut dari yang terbesar sampai yang terkecil, menggunakan perintah sebagai berikut :

```
if($i==0){
   $trSQL ="select * from useronline where room='$room'
      order by score desc" ;
```

```
$rlt = mysql_query($trSQL);
$rw = mySQL_fetch_row($rlt);
```

Sebelum memutuskan pemenangnya, pengecekan dilakukan pada nilai terbesar pertama dan kedua. Hal ini digunakan untuk mengetahui adanya keputusan permainan imbang atau draw. Bila nilai terbesar 1 dan 2 jumlahnya sama, maka diputuskan tidak ada pemenang (seri) untuk permainan di ronde ini. Selanjutnya nilai record draw pada masing-masing *user* di tabel *login* pada *database* akan diupdate (nilainya ditambahkan)

```
// pengambilan nilai terbesar pertama dan kedua
   $winner1 = $rw[3]; $winname = $rw[0];
   $rw = mySQL_fetch_row($rlt);
   $winner2 = $rw[3];

   if ($winner1 == $winner2)
      { draw ="select draw from login where name ='$name'" ;
      $drawres = mysql_query($draw);
      $re = mySQL_fetch_row($drawres);
      $new_draw = $re[0]+1;
   $stat = "update login set win = '$new_draw' where
       name = '$name'";
   print ("drawgm"); }
```

Bila tidak dijumpai hasil draw, dilakukan pengurutan nilai score, didapatkan nama *user* dengan nilai terbesar. Nama pemenang tersebut nantinya digunakan untuk mencocokan nama *user* dengan nama *user* yang me*request*nya. bila sama maka *user* tersebutlah pemenangnya.

```
if ($name == $winname )
      {$win ="select win from login where name ='$name'" ;
      $winres = mysql_query($win);
      $rw = mySQL_fetch_row($winres);
      $new_win = $rw[0]+1;
// update nilai reco win ditambah satu di database user
      $stat = "update login set win = '$new_win'
         where name = '$name'";
   print("winwin");}
```

Namun bila nama *user* dengan nilai tertinggi tidak sama dengan nama *user* yang melakukan *request*, maka *user* tersebut dinyatakan kalah dalam permainan. Proses ini diikuti dengan peng-*update*-an nilai lose di *database user* dimana nilai lose-nya akan ditambahkan.

```
//bila namanya bukan nama user maka dianggap kalah
   { $lose ="select lose from login where name ='$name'" ;
   $loseres = mysql_query($lose);
   $rx = mySQL_fetch_row($loseres);
```

```
// update nilai field lose ditambah satu di database user
      $stat = "update login set lose = '$rx[0]+1'
         where name = '$name'";
   $upd_result = mysql_query($stat);
   print("luslus");
```

Setelah dilakukan update data milik *user* yang menyangkut hasil permainan (win, lose, draw) selanjutnya pada tabel *useronline* dilakukan pengesetan nilai awal (reset) dimana status permainan diubah menjadi tidak aktif lagi dengan memberikan nilai nol pada *database*.

```
$reset = "update useronline set play = '0'
   where room = '$room'";
$upd_result = mysql_query($reset);
```

Agar memudahkan *user* untuk mengetahui siapa saja yang belum memasukkan nilai permainan ke *server* dan besar nilai masing-masing *user*, dibagian bawah layar akan ditampilkan daftar susunan nilai pemain dari terbesar sampai terkecil yang diterima oleh *server*.

```
$wiSQL="select * from useronline where room = '$room'
   order by score desc";
$res = mysql_query($wiSQL);
while ($re = mySQL_fetch_row($res))
   { $naplyr[$i] = $re[0];
   $scrplyr[$i] = $re[3];
   $i ++;}
print("++ Score pada Room ".$room." ++\n");
for ($a=0 ; $a<$i ;  $a++) {
   print ($naplyr[$a]."   ");
   if($scrplyr[$a]== -1) {    print(">  not yet\n"); }
   else { print (">  ".$scrplyr[$a]."\n"); }}
```

4.1.9. ‘quit*room*.php’

File ‘quit*room*.php’ digunakan saat *user* melakukan proses keluar *room*. Ketika *user* berada di salah satu *room*, maka *handphone* akan menampilkan tampilan menu yang di dalamnya terdapat pilihan untuk keluar *room*. Saat pilihan tersebut dipilih *user*, *handphone* akan melakukan *request* ke *server* (pada *file* ini) agar status *room user* dihilangkan sehingga *user* tidak *join* lagi dengan *room* tersebut. Berikut akan diberikan isi program secara keseluruhan :

```
$i=0;
$room="select name from useronline where room = '$room'
   and play = '1'" ;
$res = mysql_query($room);
while ($row = mySQL_fetch_row($res))
   { $i++;  }

// bila status permainan masih aktif, user tidak bisa
```

```
   keluar room
if ($i>0)
   { print ("someoneplay"); }
else {

// bila tidak aktif, reset nilai, room dan score pada user
   $roomte="update useronline set room = '0', play = '0',
      score = '-1' where name = '$name'";
   $upd_result = mysql_query($roomte);
   $newroom = 0;
   print($newroom); }
```

4.1.10. ‘chgpass.php’

Semua *user* diberikan fasilitas untuk mengubah nilai *password*, yang digunakan ketika *login*. Proses penggantian *password* cukup sederhana yaitu dengan mencocokkan nama dan *password* lama pada *user* yang bersangkutan, menggunakan perintah sebagai berikut :

```
$strSQL="select * from login where name = '$name' AND
    password='$password'";
$result = mysql_query($strSQL,$conn);
$array = mySQL_fetch_array($result);
```

Bila pasangan nama dan *password* yang dimasukkan sesuai dengan yang ada di *database*, maka dilakukan update *password* yang lama digantikan dengan nilai *password* yang baru.

```
$update="update login set password = '$newpass' where
   name = '$name' AND password='$password'";
$upd_result = mysql_query($update);
print("Password successfully change");}
```

4.1.11. ‘*logout*.php’

Ketika seorang *user* memutuskan untuk berhenti bermain dan keluar dari aplikasi, sebelum aplikasi MIDlet dihapus (destroy), maka akan dilakukan pemanggilan fungsi yang bertugas mengirimkan *request* ke *server* untuk proses *logout*. Proses yang terjadi berkebalikan dengan proses *login* dimana akan dilakukan penghapusan data *user* (name, status permainan, *room*, dan score) dari *database* table *useronline* sehingga *user* tersebut tidak dianggap sedang *online*.

```
\\ mencari data sesuai nama user
$cek_SQL="select * from useronline where name = '$name'";
$cek_sult= mysql_query($cek_SQL);
$cek_row = mySQL_fetch_array($cek_sult);

\\ bila ada, maka data tersebut dihapus
if ($cek_row>0)
```

```
{ $out_SQL= "delete from useronline where name =
   '$n   ame'";
$out_sult= mysql_query($out_SQL); }
```

4.2. Implementasi Sistem pada Sisi *Client*

Untuk implementasi sistem pada sisi *client* digunakan program Java 2 Micro Edition. Dalam hal ini dibuat sebuah program MIDlet yaitu MIDlet 'Dadu*Game*.java' dimana seluruh proses yang ditampilkan di *client* mulai dari tampilan menu, tampilan *game*, interaksi dengan *user*, penanganan jaringan (networking) ke *server* dan fungsi lainnya, dibuat di dalam program ini. Baik di sisi *client* maupun di *server*, kegiatan yang dilakukan merupakan pasangan interaksi antara *client* dengan *server*.

Berikut akan dijelaskan bagian-bagian yang penting dari sistem di *client*:

4.2.1. Inisialisasi awal 'Dadu*Game*.java'

Sebelum menjalankan proses yang ada pada sistem, perlu dideklarasikan package-package yang perlu digunakan MIDlet tersebut. *Package* yang digunakan pada *game* ini, antara lain :

```
import java.io.*;
import java.lang.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import java.util.*;
```

Selanjutnya dilakukan deklarasi class Dadu*game* yang merupakan turunan class MIDlet. Class tersebut mengimplementasikan CommandListener untuk penanganan tombol / perintah command)

```
public class Dadu Game extends MIDlet
   implements CommandListener
```

Pada bagian berikutnya, didefinisikan beberapa variable yang digunakan pada program secara keseluruhan (global) dan dilanjutkan dengan deklarasi konstruktor Dadu*Game*().

```
private DaduAnimCover cover;
private DaduAnimPlay daPlay;
private List mainMenu,accMenu,gamenu;
...

...
private String userReport = "nouser", scrResult=" ",
   myRoom="notyet",soNumb,reset=" ";
private int plyTurn=0, plyScore=0, sumScore=0,updateTime=0,
   roomNow,detik=1700;
```

Deklarasi Konstruktor yang dipakai seperti berikut :

```
public Dadu Game() {
   cover          = new DaduAnimCover(this);
   exitCmd        = new Command("Exit",   Command.OK,0);
   loginCmd       = new Command("Login",  Command.OK,0);
   ...

   againCmd       = new Command("Setuju",Command.OK,0);
   noAgainCmd     = new Command("Tidak",Command.OK,0);}
```

Selain itu, karena kelas ini mengimplementasikan `CommandListener` perlu dibuat fungsi yang digunakan untuk menangani masalah event jika terdapat tombol atau menu yang dipilih oleh *user*.

```
public void commandAction(Command c, Displayable d){
if (d == gamenu) {
if (myRoom.equals("hasinroom"))
   {  if (gamenu.isSelected(0)){ checkPlyr(); }
      else
      if (gamenu.isSelected(1)){  lookingRoom(); }
      else
      if (gamenu.isSelected(2)){ quitRoom();  }
      else
      if (gamenu.isSelected(3)){ howTo(); }
...

else
      if (gamenu.isSelected(2)) { menuForm(); }
   } }
```

4.2.2. Proses *Login* dan Tampilan Menu.

Untuk dapat melakukan pengaksesan aplikasi ini, yang pertama kali harus dilakukan oleh seorang pemain adalah melakukan proses *login* terlebih dahulu. *Username* dan *password* yang dimasukkan haruslah sesuai dengan *username* dan *password* yang telah ada di *database server*. Karena ditampilkannya pertama kali, maka fungsi yang digunakan pada proses *login* ini adalah fungsi ‘startApp()’.

Fungsi ‘startApp()’ merupakan salah satu dari tiga fungsi abstrak yang harus diimplementasikan untuk setiap MIDlet. Dalam hal ini, fungsi ‘startApp()‘ akan menjadi fungsi yang secara *default* ditampilkan saat MIDlet dijalankan pertama kali. Adapun list programnya sebagai berikut :

```
public void startApp()
   { display = Display.getDisplay(this);
   fLogin = new Form("Login Permainan");
   tUserName = new TextField("Nama Pemain : ",""
        ,20,TextField.ANY);
   tPassword = new TextField("Password    : ",""
        ,20,TextField.PASSWORD);
```

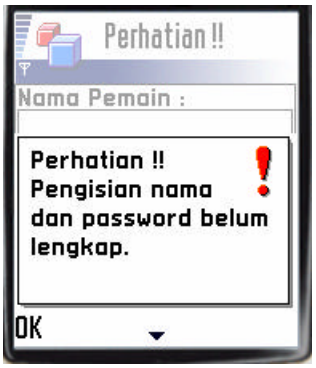
```
      fLogin.append(tUserName);
      fLogin.append(tPassword);
      fLogin.addCommand(loginCmd);
      fLogin.addCommand(exitCmd);
      fLogin.setCommandListener(this);
      display.setCurrent(fLogin);      }

// memanggil form login
      public void loginForm() {
      display.setCurrent(fLogin); }

      public void commandAction(Command c, Displayable d){
      ...
      else if (c == loginCmd)
         { usLogin(); }
...
```

Pada form fLogin di atas, terdiri dari 2 textfield dan 2 button command. Textfield tersebut masing-masing digunakan untuk inputan nama *user* dan *password*. Tampilan program bila dijalankan di *handphone* akan tampak seperti pada gambar 4.1. di bawah ini.

Gambar 4.1. Tampilan Menu *Login*

Bila button command '*Login*' dipilih (command *login*Cmd), maka dilakukan pengecekan pada aplikasi *handphone* apakah kedua field ada yang kosong. Bila terdapat kosong baik salah satu maupun semua field, maka akan dicetak peringatan berupa Alert ke layar.

```
fInfo = new Form ("Login Diproses");
if(tUserName.getString().trim().equals("")
      ||tPassword.getString().trim().equals("")){
   Alert notComplete = new Alert("Perhatian!!",
      "Pengisian nama dan password belum
      lengkap.",null,AlertType.WARNING);
   Display display = Display.getDisplay(this);
   display.setCurrent(notComplete, fLogin); }
```



Gambar 4.2. Tampilan peringatan (alert) saat *login*

Bila semua field terisi lengkap, maka dilakukan pengiriman nilai nama dan *password* tersebut ke web *server* untuk dilihat kecocokannya dengan yang telah ada di *database server*. Pengiriman ke *server* menggunakan HTTP Connection ke alamat IP *server*. Dimulai dengan membuka koneksi menggunakan `Connector.open` yang diikuti dengan alamat *file server* dan nilai variabel (nama serta *password*) yang digunakan sebagai pembanding nilai saat pengecekan data di *database*.

```
c =(HttpConnection)Connector.open("http://202.43.253.54/
   gameserver/login.php?name=" + tUserName.getString()+
   "&password=" + tPassword.getString());
if (is!= null) { is.close(); }
if (c!= null) { c.close(); } }
```

Metode pengiriman seperti yang digunakan di atas ini, banyak digunakan saat *handphone* melakukan komunikasi data dengan jaringan (*server*). Nilai balik (respon) dari *server*, diletakkan pada variabel `is`. Nilai yang ada pada `is` masih merupakan input stream. Agar bisa ditampilkan ke bentuk string, dilakukan melalui pembacaan keseluruhan byte data pada input stream melalui metode `read()`. Metode `read()` sendiri akan bekerja berulang-ulang ampai akhir data stream `is` yang ditunjukkan dengan nilai sama dengan -1.

```
is = c.openInputStream();
int ch;
while ((ch = is.read()) != -1) {
   stringBuffer.append ((char) ch); }
fromServer = new String (stringBuffer.toString());
```

Bila nama dan *password* yang tadi dikirimkan ke *server* benar, hasil respon dari *file* '*login*.php' menunjukkan hasil "*user*100%*password*". Bila yang terjadi demikian, maka `enterGame()` akan dijalankan. `EnterGame()` sendiri

nantinya akan memanggil suatu `class DaduAnimCover` yang merupakan turunan dari kelas Canvas untuk menampilkan splash screen *game*.

```
      if (from Server.equals("user100%password"))
            { enter Game();
      else { Form forms = new login Form();
            display.setCurrent(forms);}

...

      public void enter Game() {
            Display.getDisplay(this).setCurrent(cover); }

...
      class DaduAnimCover extends Canvas implements
            CommandListener {
      private Display display;
      private final Dadu Game midlet;
      private final Command continueCmd;
      private int frameHeight, frameWidth;
      private Image coverImages = null;

      DaduAnimCover(Dadu Game midlet){
            this.midlet = midlet;
            if (coverImages == null) {
            try {coverImages =      Image.createImage
                  ("/background.png");
                  frameHeight = coverImages.getHeight();
                  frameWidth = coverImages.getWidth();}
            catch (Exception ioe) { } }
            continueCmd = new Command("Masuk", Command.SCREEN,1);
            addCommand(continueCmd);
            setCommandListener(this); }

      public void commandAction(Command c, Displayable d)
            { if(c == continueCmd) { midlet.menuForm(); }}
      public void paint(Graphics g)
            { g.setColor(0xFFFFFF);
            g.fillRect(0, 0, getWidth(), getHeight());
            g.clipRect(0, 0, getWidth(), getHeight());
            g.drawImage(coverImages, (getWidth()  - frameWidth)/2,
            (getHeight()-frameHeight)/2, Graphics.TOP +
            Graphics.LEFT); } }

      ...

      if(c == continueCmd) { midlet.menuForm(); }

      ...
```

Gambar 4.3. Tampilan *splash screen*

Bila *user* sudah memilih pilihan 'Masuk' saat di splash screen, maka akan ditampilkan `menuForm()` yang terdiri dari 4 pilihan yaitu menu Permainan, menu Account, menu Help dan Menu Keluar.

Gambar 4.4. Tampilan Menu Awal

Bila *user* belum terdaftar di suatu *room*, bila memilih menu Permainan, menu akan menampilkan 3 pilihan, yaitu Cek *Room*, Cara Bermain dan Kembali ke menu awal. Namun bila *user* telah *join* dengan suatu *room* maka akan muncul 4 pilihan menu yaitu : Play!, Cek *Room*, Keluar *Room*, dan Cara Bermain. Untuk bisa mengetahui apakah seorang pemain sudah mendapat *room* atau belum digunakan variabel my*Room*, bila nilainya sama dengan 'hasin*room*', berarti *user* tersebut sudah terdaftar dengan suatu *room*.

```
public void gameMenu(){
   if (myRoom.equals("hasinroom"))
      {userbanner=new String("Anda Join Room "+ soNumb);}
   else { userbanner = new String("Let's Join Room"); }

   gamenu=new List(userbanner , List.IMPLICIT);
   if (myRoom.equals("hasinroom"))
      {gamenu.append(" Play !", null);}
   gamenu.append(" Cek Room", null);
   if (myRoom.equals("hasinroom"))
```

```
      {gamenu.append(" Keluar Room", null);}
   gamenu.append(" Cara Bermain", null);
   if (myRoom.equals("notyet"))
      {gamenu.append(" Kembali", null); }
   gamenu.setCommandListener(this);
   display.setCurrent(gamenu); }
```

Gambar 4.5. Tampilan Menu bila *User* Belum dan Telah *Join Room*

4.2.3. Proses *Join Room*

Pada permulaan ketika ingin bermain, *user* harus *join* dahulu dengan *room* sehingga *user* harus memilih pilihan Permainan dari menu awal dan dilanjutkan memilih Cek *Room*. Ketika pilihan dieksekusi, terjadi proses komunikasi ke *server* untuk mendapatkan data terbaru tentang pemakai *room* saat itu. Proses yang digunakan untuk mendapatkan informasi tersebut mirip pada saat proses *login* dilakukan.

```
HttpConnection c = null;
InputStream is = null;
StringBuffer stringBuffer = new StringBuffer();
try
   {c = HttpConnection)Connector.open("http://202.43.253.54/
   gameserver/seeroom.php?name=" + tUserName.getString());
   is = c.openInputStream();
   int ch;
   while ((ch = is.read()) != -1)
      { stringBuffer.append((char) ch); }
   userReport = new String (stringBuffer.toString());
if (userReport.equals("nouser"))
   { title = "Checking User";
   userReport = " Belum ada user yang Online
      sekarang. Cobalah beberapa saat lagi";
   Form NoOnline = new inforForm();
   display.setCurrent(NoOnline); }
else
   { Form formCheck = new formInvite();
   display.setCurrent(formCheck); }
   if (is!= null) { is.close(); }
```

```
   if (c!= null) {      c.close(); } }
catch (IOException e) { } }
```

Gambar 4.6. Tampilan Cek *Room*

Setelah nilai *room* dimasukkan, *user* akan diberikan informasi berhasil atau tidaknya proses *join room* yang baru dilakukan. Proses pengiriman *request* sama dengan proses networking yang sudah dilakukan sebelumnya (ketika *login* dan melihat data *room*), hanya *file* PHP yang dituju di *server* adalah *file* '*joinroom*.php'.

```
public void db Room() {
   current Room = soNumb;
   HttpConnection c = null;
   InputStream is = null;
   StringBuffer stringBuffer = new StringBuffer();
   try {
      c = (HttpConnection)Connector.open
         ("http://localhost/gameserver/joinroom.php?name=
         " + t UserName.getString() + "&room=" + soNumb);
      is = c.openInputStream();
      int ch;
   while ((ch = is.read()) != -1)
      { stringBuffer.append((char) ch); }
   roomReport = new String (stringBuffer.toString());
```

Ketika konfirmasi dari *server* menyebutkan bahwa *user* berhasil *join* dengan *room*, maka secara otomatis, tampilan akan diubah ke *waiting room*. Di dalam *waiting room*, ditampilkan data jumlah *user* yang sudah *join* pada *room* tersebut. Data *room* ini akan diupdate setiap interval waktu yang ditentukan.

Pengupdatean data ini menggunakan kelas timer sebagai penghitung waktunya, dimana memiliki sifat sebagai bacground thread sehingga Bila terdapat lebih dari satu orang di *room*, maka pilihan 'PlayNow' akan muncul.

```
   ...
   if (roomReport.equals("success"))
      {
      userReport = " Join ke Room " + soNumb + " berhasil.
         Pengecekan server dilakukan.\nPlease wait!";
      updateTime = 1000;
      Form rmRdy = new WaitJoin(this);
      display.setCurrent(rmRdy);
      myRoom = new String("hasinroom"); }
...

...
class WaitJoin extends Form {
   private Display display;
   private String passMsg;
   private DaduGame daduGame;

WaitJoin(DaduGame daduGame) {
   super("Waiting Room");
   waitTime = new Timer();
   TimerTask wait = new waitTime();
   TimerTask wait2 = new waitTime2();
   this.daduGame=daduGame;
   passMsg = userReport;
   append(passMsg);
   sTick();

   waitTime.schedule(wait,updateTime);
   waitTime.schedule(wait2,1000,1000);
   if (unablePlay.equals("no"))
      { addCommand(playCmd); }
   addCommand(backGameCmd);
   setCommandListener(DaduGame.this); }

...
```

Gambar 4.7. Tampilan *Waiting Room*

4.2.4. Proses Tampilan Permainan

Permainan bisa dimulai oleh *server* maupun oleh *client*. *Server* akan mengaktifkan status permainan bila *user* di dalam suatu *room* dianggap penuh yaitu 4 *user*, yaitu mengubah status permainan *room* menjadi 1.

User bisa pula memulai permainan bila di dalam *room* terdapat lebih dari 1 orang pemain, yang ditunjukkan dengan munculnya pilihan 'PlayNow' seperti yang ditunjukkan pada gambar 4.6. Bila terjadi pemilihan opsi tersebut, maka status permainan di *database* pada *user* yang *room*nya sama akan diubah menjadi aktif atau bernilai satu. Bila saat *handphone* me-refresh data ke *server* (ketika di *waiting room*) dan responnya diterima, maka akan tampil informasi bahwa permainan akan dimulai. Tampilan di layar akan nampak seperti gambar 4.7.

Gambar 4.8. Tampilan Layar Permainan

Penanganan grafik pada pembuatan *game* ini menggunakan kelas dari Canvas yang mengimplementasikan CommandListener dan Runnable. Penggunaan ComandListener karena kerja sistem dapat diinterupsi oleh *user* melalui interaksi menggunakan tombol. Implementasi Runnable digunakan karena Thread digunakan ketika sistem dijalankan. Thread akan berhenti saat *user* menginterupsi atau kondisi false terjadi.

```
...
public class DaduAnimPlay extends Canvas
   implements CommandListener, Runnable {

...
   private boolean running=false;
   goCmd = new Command("Kocok",Command.SCREEN,3);
   stopCmd = new Command("Stop",Command.SCREEN,3);

...
```

```
public void run() {
try { while (running) {
   Thread.sleep((frameIndex == FRAME_COUNT - 1)
      ? LAST_FRAME_DELAY : FRAME_DELAY);
   int lastFrameIndex  = frameIndex;
   frameIndex = (frameIndex + 1) % FRAME_COUNT;
   count = randRange(6);
   showLoop =  ("Giliran - " + (new Integer(plyTurn));
   showClock = ((new Integer(remain)).toString());
   showScore = ("nilai = " + scrResult);
   repaint();
} } catch (InterruptedException e) {} }
...

if(c == goCmd)
   { removeCommand(goCmd);
   addCommand(stopCmd);
   setCommandListener(this);
   //Thread start
   DaduMix(); }
if(c == stopCmd){
   //Thread stop
   running = false;
   timer.cancel();
   plyScore = count + count2;
   midlet.SelectDone(plyScore); } }
...
```

Pengaturan tampilan *Game* ditangani oleh kelas Canvas bersama-sama dengan kelas turunannya yaitu kelas Graphics untuk fungsi pencetakan gambar dan tulisan ke layar.

```
...
public void paint(Graphics g) {
   g.setColor(0xFFFFFF);
   g.fillRect(0, 0, getWidth(), getHeight());
   g.setColor(0,0,0);
   g.drawImage(gambar[count], ((getWidth()/3)-15), 30,
      Graphics.TOP + Graphics.LEFT);
   g.drawImage(gambar[count2],((getWidth()/3*2)-15),30,
      Graphics.TOP + Graphics.LEFT);
   g.setFont(Font.getFont(Font.FACE_PROPORTIONAL,
      Font.STYLE_PLAIN,Font.SIZE_SMALL));
   g.drawString(showClock,10,10,
      Graphics.TOP + Graphics.LEFT);
   g.drawString(showLoop,10,70,
      Graphics.TOP + Graphics.LEFT);
   g.drawString(showScore,10,80,
      Graphics.TOP + Graphics.LEFT);}
...
```

Nilai diambil dari penjumlahan angka dua buah dadu tersebut sesaat setelah *user* melakukan penekanan tombol ’Stop’. Setiap penekanan tombol ’Stop’ dibatasi kurang dari 5 detik setiap setelah pengacakan dadu dilakukan. Pengacakan dan pengambilan nilai dilakukan sebanyak tiga kali dalam satu

permainan dimana setiap berhasil melakukan pengambilan nilai, *user* akan diberikan informasi nilai yang didapatkan.

```
//tampilan konfirmasi nilai
Show (Dadu Game midlet, int plyScore) {
   super("Nilai Anda ");
   this.midlet = midlet;
   timer.schedule(task,detik);
   append(" Anda mendapat nilai " + (new Integer(plyScore))
      .toString() + " pada giliran ke " + (new Integer
      (plyTurn)).toString() + ".\nAkumulasi nilai : "
      + scrResult + " = " + (new Integer(sumScore))
      .toString()); }

class exTask extends TimerTask {
   public void run() {
      plyTurn ++ ;
// bila belum 3 kali bermain, ulangi lagi,
//bila sudah kirim nilai ke server
      if (plyTurn <= 3){ midlet.DaduAnimCoverDone();}
      else  { midlet.checkScr(); } }
```

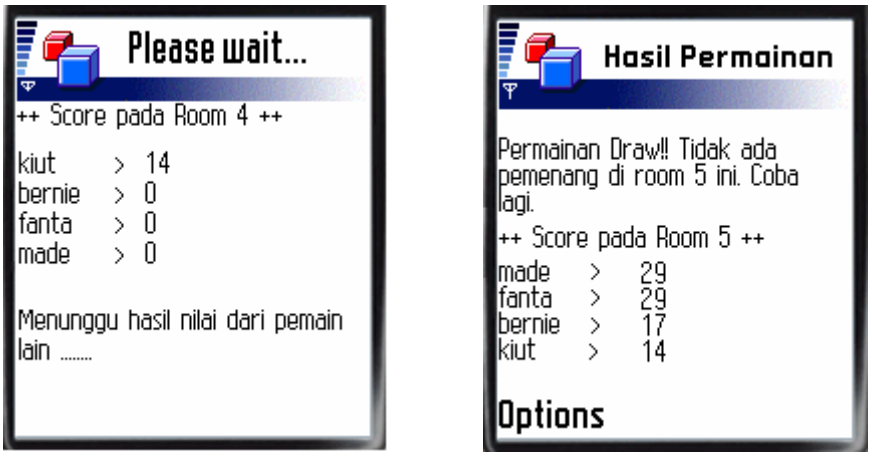
Gambar 4.9. Tampilan Konfirmasi Nilai

4.2.5. Proses Pengiriman Nilai dan Pengecekan Hasil Permainan

Pengambilan nilai diulang tiap *user* sebanyak 3 kali, setelah itu hasil penjumlahan 3 nilai tadi diinformasikan ke *user* dan dilakukan pengiriman nilai ke *server*. Pengiriman nilai ke *server* menggunakan cara sama dengan proses komunikasi pada proses sebelumnya (saat *login* dan pengecekan *room*) dimana menggunakan `HttpConnection`, hanya saja nama *file* yang dituju berbeda.

```
...
c = (HttpConnection)Connector.open("http://202.43.253.54/
   gameserver/sendscore.php?name=" + tUserName.getString()
   +"&room=" + soNumb + "&score=" + (new Integer(sumScore)
   .toString()));
...
```

Setelah pengiriman nilai *user* ke *database* di *server*, proses selanjutnya yang dilakukan sistem adalah melakukan pengecekan nilai *user* lain di *database* tersebut. Pengecekan ke *database* dilakukan secara periodik oleh *handphone.* Selama ada *user* lain yang belum mengirimkan nilai ke *server*, pengolahan hasil permainan tidak dapat dilakukan. Tampilan di layar adalah klasemen sementara dari nilai *user* yang sudah masuk ke *server*.



Gambar 4.10. Tampilan Klasemen Sementara dan Hasil Akhir

Proses pengecekan (refresh data) akan berhenti bila ada respon dari *server* yang menunjukkan hasil permainan (menang, kalah, atau draw). *Handphone* selanjutnya akan memberikan informasi akhir apakah *user* yang bersangkutan menag, kalah atau permainannya draw. Selanjutnya permainan dapat diulangi lagi ataupun tidak sesuai kehendak *user.*

4.2.6. Proses Penggantian *Password*

Pada menu Account, terdapat pilihan untuk mengganti nilai *password* yang digunakan sekarang. Tampilannya terdiri dari 3 field yang harus diisi semuanya. '*Password* Lama' nantinya diisikan *password* yang terakhir digunakan oleh *user*. '*Password* Baru' dan 'Konfirmasi' diisikan dengan *password* yang ingin digunakan menggantikan *password* yang lama. Untuk '*Password* Baru' dan 'Konfirmasi', harus diisi dengan karakter yang sama persis penggunaan hurufnya baik huruf besar maupun kecil karena sifatnya *case sensitive.*

Bila keduanya sama, maka selanjutnya *handphone* akan me*request server* untuk mengupdate data *password* pada *user* name yang bersangkutan. Bila tidak,

maka akan muncul konfirmasi bahwa *password* yang dimasukkan tidak sama. Setelah respon dari *server* diterima, akan ditampilkan informasi apakah *password* berhasil diganti atau tidak.

```
      ...
      public void changePass() {
         display = Display.getDisplay(this);
         fChange = new Form("Ubah Password");
         tPass = new TextField("Password Lama: ","",150,TextField
            .PASSWORD);

         tNewPass = new TextField("Password Baru: ","",100,
            TextField.PASSWORD);
         tNewConfirm = new TextField("Konfirmasi   : ","",100,
            TextField.PASSWORD);
         fChange.append(tPass);
         fChange.append(tNewPass);
         fChange.append(tNewConfirm);
         fChange.addCommand(changeCmd);
         fChange.addCommand(menuPasswordCmd);
         fChange.setCommandListener(this);
         display.setCurrent(fChange);}
...

      public void usLogin()
         { HttpConnection c = null;
         InputStream is = null;
         StringBuffer stringBuffer = new StringBuffer();
         fInfo = new Form ("Login Diproses");
         if(tUserName.getString().trim().equals("")||
            tPassword.getString().trim().equals(""))
            {Alert notComplete = new Alert("Perhatian
            !!","Pengisian nama dan password belum lengkap.
            ",null,AlertType.WARNING);
         Display display = Display.getDisplay(this);
         display.setCurrent(notComplete, fLogin); }
      else
         {try{c = (HttpConnection)Connector.open
            ("http://localhost/gameserver/login.php?name=" +
      tUserName.getString() + "&password=" + tPassword.
         getString());
      is = c.openInputStream();
      int ch;
      while ((ch = is.read()) != -1) { stringBuffer.
         append((char) ch); }
      fromServer = new String (stringBuffer.toString());
      if (fromServer.equals("user100%password")) { enterGame(); }
      else { Form forms = new loginForm();
               display.setCurrent(forms);}

...
```

Gambar 4.11. Tampilan Penggantian *Password*

Setelah file keseluruhan isi 'DaduGame.java' disimpan, menggunakan Java 2 *Micro Edition Wireless Tool Kit* (J2MEWTK), dilakukan proses *preferify* dan *compile file* java. Sebelum *compile* dilakukan, J2MEWTK akan melakukan pengecekan terhadap *script* program tersebut bila terdapat *error* atau kesalahan pada *script* program maka akan muncul pesan error pada console J2MEWTKdan *compile* tidak dapat dilakukan.

Bila tidak terdapat kesalahan program, compile yang dilakukan J2MEWTK akan menghasilkan *class-class* Java dan ada pesan di *console* yang menunjukkan bahwa *compile* berhasil dilakukan. Melalui operasi ini, aplikasi sudah dapat dijalankan menggunakan *emulator* pada komputer.

Operasi *packaging* dilakukan sebagai langkah terakhir yang dilakukan bila aplikasi yang kita buat, akan dijalankan ke *handphone*. Aplikasi MIDlet yang ada akan dibungkus menjadi file *.jad dan *.jad setelah *packaging* berhasil dilakukan. Nantinya kedua file tersebut (DaduGame.jad dan DaduGame.jar) dipindahkan ke handphone untuk menjalankan aplikasi dari *handphone*. Cara pemindahan dan proses instalasi pada handphone disesuaikan dengan tipe dan merek handphone yang bersangkutan.