BAB IV

IMPLEMENTASI DAN PEMBAHASAN SISTEM

4.1 Implementasi Sistem

Implementasi sistem adalah tahap untuk menjelaskan proses dalam program. Sebelum program di implementasikan, maka program harus bebas dari kesalahan agar program dapat berjalan sesuai dengan yang diharapkan. Kesalahan program yang mungkin terjadi antara lain karena kesalahan penulisan (*coding*), kesalahan proses, atau kesalahan logika.

Dalam tahap implementasi aplikasi permainan *"Hero's Key"* ini, analisis kebutuhan perangkat pendukung menjadi hal yang sangat penting. Sistem ini dapat berjalan dengan baik apabila memenuhi standar minimal dari pernagkat keras (*hardware*) yang telah ditetapkan sebelumnya dalam tahap analisis kebutuhan sistem. Selain itu kebutuhan perangkat lunak (*software*) juga harus tersedia demi kelancaran tahap implementasi program.

Dalam proses mengimplentasi program permainan *Hero's Key* ini ada beberapa langkah-langkah yang dilakukan, yaitu :

1. Menuliskan kode program (*coding*), tahap ini dilakukan dengan menggunakan program pengembang aplikasi *Java*, yaitu *Gel 1.0.0.0.*

2. Melakukan proses *compile* dengan menggunakan fasilitas yang disediakan oleh *Gel 1.0.0.0*.
3. Menguji (*run*) program dengan menggunakan fasilitas yang disediakan oleh *Gel 1.0.0.0*.
4. Analisis jalannya program dalam perangkat yang sesungguhnya, serta melakukan *debugging* atau perbaikan program jika diperlukan.

4.2 Implementasi Pengkodean (*Coding*)

Pada tahap pengkodean, secara garis besar pembahasan program permainan ini dapat dibagi menjadi tiga bagian, yaitu : program utama, menu *game*, dan isi *game*.

4.2.1 Program Utama

Program utama dari permainan ini merupakan `class dian()` yang mendefinisikan deklarasi *class* dengan menuruni sifat dari *Jframe* dengan mengimplementasikan sebuah *action*. Kelas ini merupakan kelas utama dalam sebuah aplikasi *J2SE*, yaitu semua kelas yang terdapat dalam aplikasi ini harus merupakan turunan dari kelas ini.

Untuk mendukung `class dian()` maka diperlukan beberapa *java API*, seperti *source code* di bawah ini :

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.*;
import javax.swing.KeyStroke;
import static java.awt.event.InputEvent.*;
```

```
import java.io.*;
import java.util.*;
```

Dalam `class dian()` terdapat beberapa *procedure* yang akan digunakan dalam aplikasi permainan *Hero's Key* ini. Pada *procedure actionperformed* memiliki beberapa *actio*n yaitu *listener new user*, *listener select user*, dan *listener start game*. Pada *listener new user*, aksi awal yang akan di jalankan adalah pada saat mengklik button submit maka *listener* akan memvalidasi nama yang telah tersimpan diregistrasikan, sepeti *source code* di bawah ini :

```
//----------------------LISTENER NEW USER------------------
else if(source==btnNUserSubmit)
{    // validasi nama
     if(txtNama.getText()==null ||
txtNama.getText().length()==0)
     {     JOptionPane.showMessageDialog(null,"Please Fill
Your Name","Information",JOptionPane.INFORMATION_MESSAGE);
           return;
     }     // menyimpan nama user
           String tempNamaUser=txtNama.getText();
           // menampung message autosifikasi
           String msg = tempNamaUser +"\n";
           if(JOptionPane.showConfirmDialog(null, msg+"Are
You Sure With These Data ?", "Information",
JOptionPane.OK_CANCEL_OPTION,JOptionPane.QUESTION_MESSAGE)=
=0)
```

Pada *listener select user*, aksi yang akan di jalankan adalah pada saat jendela *select user* tampil, jika *user*nya kosong akan tampil pesan peringatan "*there are no user*", tetapi apabila ada *user* maka pemain tinggal memilih usernya dan mengklik button select dan ok. Setelah mengklik ok maka akan langsung tampil *game* level 1, seperti *source code* ini :

```
//------LISTENER SELECT USER ---------------------------
else if(source==btnSUserSelect)
```

```
{    if(listUser.getSelectedIndex()==-1 )
     {     JOptionPane.showMessageDialog(null, "There Are
No User", "Warning", JOptionPane.ERROR_MESSAGE);
     }     else if(JOptionPane.showConfirmDialog(null, "Are
You Sure "+listUser.getSelectedValue()+" ?", "Information",
JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE)==0)
     {     windowSelectUsr.dispose();
           PlayedUser =
listUser.getSelectedValue().toString();
// GAME DIMULAI.........................
// munculin Playgame,Scoring, dan window start game
     StartOn();
//--------- GAME DIMULAI......................
     }}   else if (source==btnSUserCancel)
     {     windowSelectUsr.dispose();
           setEnabled(true);
           show();     }
```

Pada *listener start game*, ada aksi keluar dari level, dengan menekan button quit maka akan secara otomatis keluar dari *game* dan kembali ke menu awal.

```
// listener start game
     else if(source==btnMScoringQuit)
     {     QuitOff();
           setEnabled(true);
           show();     }
```

4.2.2 Awal Game

Pada awal *game* dari aplikasi *game Hero's Key* ini terdapat bermacam-macam menu seperti menu *new game*, menu *registration*, menu *score* dan menu *exit*. Dalam menu *game* terdapat jendela *select user* yang otomatis keluar apabila memilih menu *new game*.

```
if(source==menuNewgame)
{    if(initialize()==1)
     {     vectorSelectUser.removeAllElements();
           for(int i=0;i<peoplesTemp.size();i++)
           {     String people = (String)
peoplesTemp.elementAt(i);
                 String split[] = people.split(";");
                 vectorSelectUser.add(split[0]);
                 listUser.updateUI();
       }   windowSelectUsr.setVisible(true);
```

```
            btnSUserSelect.setVisible(true);
            btnSUserCancel.setVisible(true);
            setEnabled(false);
            windowSelectUsr.show();}}
```

Pada menu *registration* akan diminta untuk meng-*input*-kan nama *user* lalu klik submit dan ok maka nama *user* telah tersimpan otomatis di `user.sav` seperti *source code* dibawah ini :

```
if(source==menuRegisNew)
{     txtNama.setText("");  // mengosongkan
      btnNUserSubmit.setVisible(true);
      btnNUserCancel.setVisible(true);
      setEnabled(false);
      winNUser.setVisible(true);
      winNUser.show();}
```

Pada menu *score* akan menampilkan 5 peringkat tertinggi *user* yang telah bermain yang datanya di ambil dari `hiScore.sav` seperti *source code* dibawah ini :

```
if(source==menuScore)
{     windowScore.setVisible(true);
      btnClose.setVisible(true);
      setEnabled(false);
      //-------------NAMPILIN HASIL DARI DATA hiScore.sav
      Vector VecHiScore = new Vector();
      try
      {     BufferedReader buf2 = new BufferedReader(new
FileReader("hiScore.sav"));
            String line2 = null;
            while((line2=buf2.readLine())!=null) {
            VecHiScore.add(line2);
      }     buf2.close();
            }catch(Exception ex) {  }
            String TempUsrx[] = new String[5];
            String TempScrx[] = new String[5];
            String TempStepx[] = new String[5];
            for(int i=0;i<VecHiScore.size();i++)
            {     String CurrentUser2 = (String)
VecHiScore.elementAt(i);
                  String split2x[] =
CurrentUser2.split(";");
                  TempUsrx[i] = split2x[0];
                  TempScrx[i] = split2x[1];
                  TempStepx[i] = split2x[2];
            }System.out.print(TempUsrx[1]+" "+TempScrx[1]+"
"+TempStepx[1]);
            int cek[] = new int[5];
```

```
            for(int i=0;i<5;i++){  cek[i]=
Integer.parseInt(TempScrx[i]);  }
            if(cek[0]!=0)
            {     lblStripKingName.setText(TempUsrx[0]);
                  lblStripKingScore.setText(TempScrx[0]);
                  lblStripKingStep.setText(TempStepx[0]);
            }else {
                  lblStripKingName.setText("---");
                  lblStripKingScore.setText("---");
                  lblStripKingStep.setText("---");
            }if(cek[1]!=0){
                  lblStripQueenName.setText(TempUsrx[1]);
                  lblStripQueenScore.setText(TempScrx[1]);
                  lblStripQueenStep.setText(TempStepx[1]);
            }else {
                  lblStripQueenName.setText("---");
                  lblStripQueenScore.setText("---");
                  lblStripQueenStep.setText("---");
            }if(cek[2]!=0){
                  lblStripKnightName.setText(TempUsrx[2]);
                  lblStripKnightScore.setText(TempScrx[2]);
                  lblStripKnightStep.setText(TempStepx[2]);
            }else {
                  lblStripKnightName.setText("---");
                  lblStripKnightScore.setText("---");
                  lblStripKnightStep.setText("---");
            }if(cek[3]!=0){
                  lblStripBishopName.setText(TempUsrx[3]);
                  lblStripBishopScore.setText(TempScrx[3]);
                  lblStripBishopStep.setText(TempStepx[3]);
            }else {
                  lblStripBishopName.setText("---");
                  lblStripBishopScore.setText("---");
                  lblStripBishopStep.setText("---");
            }if(cek[4]!=0){
                  lblStripSquireName.setText(TempUsrx[4]);
                  lblStripSquireScore.setText(TempScrx[4]);
                  lblStripSquireStep.setText(TempStepx[4]);
            }else {
                  lblStripSquireName.setText("---");
                  lblStripSquireScore.setText("---");
                  lblStripSquireStep.setText("---");
            }//-- END NAMPILIN HASIL DARI DATA hiScore.sav
      windowScore.show();}
```

4.2.3 Isi Game

Pemrograman pada bagian ini merupakan inti dari pembuatan program *game*. Pada bagian ini berisi program untuk menjelaskan level *game* yang telah ditentukan. Pada

permainan level 1 di mulai dengan langkah awal sebanyak 50 langkah. Apabila pemain berhasil membuka gembok dengan kunci yang telah tersedia sebelum langkahnya habis maka permainan selesai dan lanjut ke level selanjutnya. Namun apabila pada level pertama ini tidak menang maka akan langsung keluar dari permainan dan kembali ke menu awal. Permainan level 2 di mulai dengan langkah awal sebanyak 60 langkah. Apabila pemain berhasil membuka gembok dengan kunci yang telah tersedia sebelum langkahnya habis maka permainan selesai dan lanjut ke level selanjutnya. Namun apabila pada level kedua ini tidak menang maka akan langsung keluar dari permainan dan kembali ke menu awal. Permainan level 3 di mulai dengan langkah awal sebanyak 70 langkah. Apabila pemain berhasil membuka gembok dengan kunci yang telah tersedia sebelum langkahnya habis maka permainan selesai dan data akan tersimpan. Namun apabila pada level ketiga ini tidak menang maka akan langsung keluar dari permainan dan kembali ke menu awal lagi. Setiap nama *user, step*, level dan *score* akan ditampilkan ditempat yang telah di tentukan. Berikut adalah *source code* :

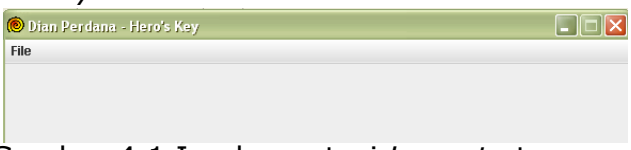
```
if(CurrLv==1)
{    LockLv1--;
     if(LockLv1==0)
     {     CurrStep=CurrStep+(50-StepLv1);
           CurrScore=CurrScore+(StepLv1*100);
           lblMScoringScoreValue.setText(""+CurrScore);
     JOptionPane.showMessageDialog(windowPlayGameLv1,"Congr
atulation,                 Level                 "+CurrLv+"
Accomplish","Message",JOptionPane.INFORMATION_MESSAGE);
```

```
            CurrLv=2;
            LoadMap(CurrLv);
            lblMScoringLevelValue.setText(""+CurrLv);
            lblMScoringStepValue.setText(""+StepLv2);
}}else if(CurrLv==2)
{    LockLv2--;
     if(LockLv2==0)
     {      CurrStep=CurrStep+(60-StepLv2);
            CurrScore=CurrScore+(StepLv2*100);
            lblMScoringScoreValue.setText(""+CurrScore);
     JOptionPane.showMessageDialog(windowPlayGameLv1,"Congr
atulation,                 Level                 "+CurrLv+"
Accomplish","Message",JOptionPane.INFORMATION_MESSAGE);
            CurrLv++;
            LoadMap(CurrLv);
            lblMScoringLevelValue.setText(""+CurrLv);
            lblMScoringStepValue.setText(""+StepLv3);
}}else if(CurrLv==3)
{    LockLv3--;
     if(LockLv3==0)
     {      CurrStep=CurrStep+(70-StepLv3);
            CurrScore=CurrScore+(StepLv3*100);
            lblMScoringScoreValue.setText(""+CurrScore);
     //=======WON WON
     JOptionPane.showMessageDialog(windowPlayGameLv1,"CONGR
ATULATION,     "+PlayedUser+"     YOU     HAVE     WON     THE
GAME","Message",JOptionPane.INFORMATION_MESSAGE);
     setEnabled(true);
```

Berikut adalah tampilan-tampilannya :

a. Implementasi *Layout* Utama

Pada saat pertama *game* di jalankan yang akan tampil pertama adalah menu *layout* utama. Berikut ini adalah tampilan *layout* utama :



Gambar 4.1 Implementasi *Layout* utama

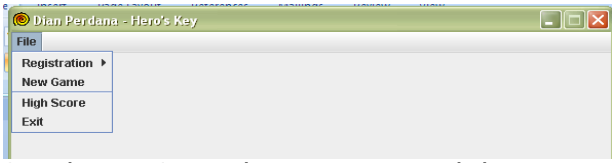
Source code dari implementasi *Layout* utama:

```
setTitle("Dian Perdana - Hero's Key");
setIconImage((new
ImageIcon("Images/founder.png")).getImage());
setBounds(90,   40,   590,   510);   //(posisi   x,posisi
y,lebar,tinggi)
// konfigurasi layout utama
c.setLayout(null); // layout panel dikosongkan
```

Pada implementasi *layout* utama, disini diberi title "Dian Perdana – Hero's Key". Ukuran awal dari layout pertama di beri nilai posisi x = 90, posisi y = 40, lebar = 590, tinggi = 510, kemudian *layout* panel di kosongkan. Pada *layout* utama di dalam menu *File* terdapat beberapa menu yaitu menu *Registration*, *New Game*, *High Score* dan *Exit*.

b. Implementasi Jendela Menu

Pada saat mengklik tab *File* maka akan tampil jendela menu. Berikut adalah tampilannya :



Gambar 4.2 Implementasi Jendela Menu

Source code dari implementasi jendela menu seperti:

```
JMenuBar menuBar = new JMenuBar();
JMenu fileMenu = new JMenu("File");
JMenu menuRegis = new JMenu("Registration");
JMenuItem menuNewgame = new JMenuItem("New Game");
JMenuItem menuScore = new JMenuItem("High Score");
JMenuItem menuExit = new JMenuItem("Exit");
```

c. Implementasi *Registration New User*

Jika ingin memainkan *game*, maka *user* harus melakukan *registration* terlebih dahulu. Berikut adalah tampilannya :

Gambar 4.3 Implementasi *Registration New User*

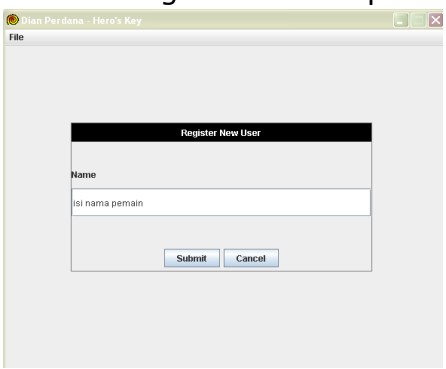
Source code seperti berikut ini :

```
// LABEL tambahan untuk Frame NEW USER
JLabel lblNUName = new JLabel("Name");
JLabel lblRNUser = new JLabel("Register New User",JLabel.CENTER);
```

Berikut adalah tampilan dari langkah-langkah *Registrasi New User* :

- *Registration New User*

  Setelah memilih menu *registration* → *new* user maka akan menghasilkan tampilan seperti berikut ini :



Gambar 4.4 *Registration New User*

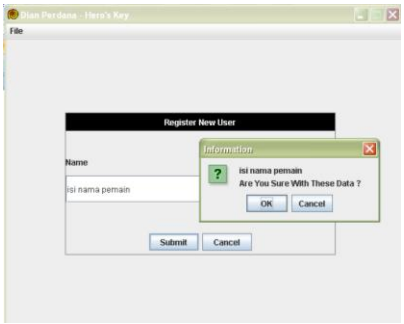
Source code dari *Registration New User* seperti :

```
txtNama.setText("");
btnNUserSubmit.setVisible(true);
btnNUserCancel.setVisible(true);
setEnabled(false);
winNUser.setVisible(true);
winNUser.show();
```

Pada saat jendela *registration new user* tampil maka akan di minta untuk meng-*input*-kan nama *user*. Setelah meng-*input*-kan nama dengan benar pilihlah button submit.

- *Message Box Registration New User*

Apabila pemain telah mengisi data *registration new user* maka pilih submit dan akan tampil message box pilihan seperti berikut ini :



Gambar 4.5 *Message Box Registration New User*

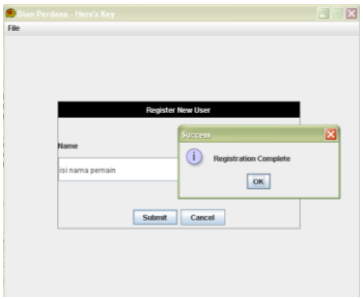
Source code seperti berikut ini :

```
(source==btnNUserSubmit)
{  // validasi nama
   if(txtNama.getText()==null||txtNama.getText().length()
==0)
   {     JOptionPane.showMessageDialog(null,"Please  Fill
Your
Name","Information",JOptionPane.INFORMATION_MESSAGE);
   return;
   }String     tempNamaUser=txtNama.getText();//menyimpan
nama user
String  msg  =  tempNamaUser  +"\n";//  menampung  message
autosifikasi
   if(JOptionPane.showConfirmDialog(null,  msg+"Are  You
Sure    With    These    Data    ?",    "Information",
JOptionPane.OK_CANCEL_OPTION,JOptionPane.QUESTION_MESSAG
E)==0)
```

Apabila mengklik button submit dan nama pemain belum di isi atau kosong maka akan muncul *message box "Please Fill Your Name"* dan kembali ke awal. Sedangkan apabila nama *user* telah di isi, nama *user* akan disimpan dan kemudian akan muncul *message box "Are You Sure With These Data?"*, jika data tersebut benar maka langsung ok.

- *Registration* Berhasil

  Setelah memilih submit maka akan tampil *message box* yang menandakan bahwa proses *registration* selesai dan kemudian klik ok. Berikut adalah tampilannya :



Gambar 4.6 *Registration* Berhasil

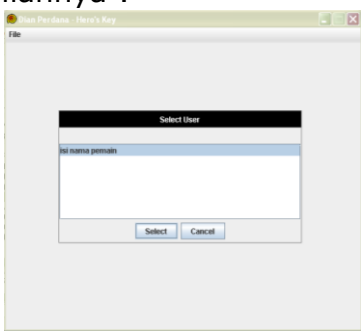
Source code dari implementasi *registration* berhasil :

```
JOptionPane.showMessageDialog(null,"Registration
Complete","Success",JOptionPane.INFORMATION_MESSAGE);
```

Jika telah melakukan registrasi dengan benar maka akan muncul *message box "Registration Complete".*

d. Implementasi Select *User*

Setelah melakukan *Registration*, permainanpun bisa di mulai dengan memilih *user* di jendela select *user*, berikut adalah tampilannya :

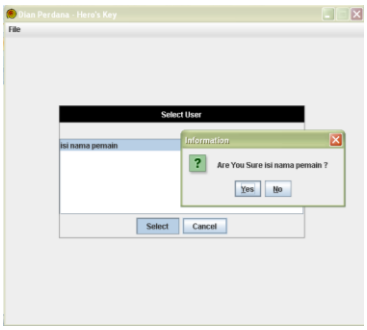


Gambar 4.7 Implementasi *Select User*

Berikut adalah tampilan langkah-langkah select user :

- *Message Box Select User*

  Apabila telah memilih *user* maka klik select maka akan keluar *message box* yang tampil seperti berikut ini :



Gambar 4.8 *Message Box Select User*

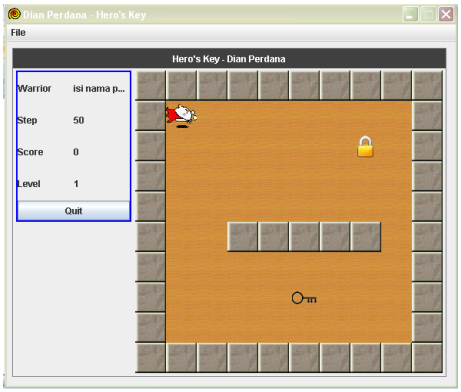
Source code dari implementasi di atas adalah:

```
if(source==btnSUserSelect)
{  if(listUser.getSelectedIndex()==-1 )
   {     JOptionPane.showMessageDialog(null,  "There  Are
No User", "Warning", JOptionPane.ERROR_MESSAGE);
   }else  if(JOptionPane.showConfirmDialog(null,  "Are  You
Sure  "+listUser.getSelectedValue()+"  ?",  "Information",
JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE)=
=0)
   {     windowSelectUsr.dispose();
   PlayedUser = listUser.getSelectedValue().toString();
```

Pada jendela *select user*, akan di tampilkan *user* yang telah teregristrasi dengan benar. Jika sewaktu membuka menu new game dan usernya belum terdaftar maka akan muncul message box *"there are no user"*. Sedangkan apabila telah terdaftar maka tinggal memilih *user* yang benar, setelah itu akan muncul message box pertanyaan *"Are You Sure (nama user) ?"*, jika data *user* benar maka langsung klik ok dan permainan akan di mulai.

e. Implementasi Level 1

Ini adalah tampilan awal permainan. Pada level pertama tingkat kesulitan untuk membuka gembok masih sedikit. Gembok pada level pertama hanya satu. Berikut ini adalah tampilan sebelum gembok terbuka:



Gambar 4.9 Implementasi Level 1

Source code dari implementasi jendela level 1 :

```
// cek level dan gembok yang sudah selesai
if(CurrLv==1)
{  LockLv1--;
   if(LockLv1==0)
   {      CurrStep=CurrStep+(50-StepLv1);
          CurrScore=CurrScore+(StepLv1*100);
          lblMScoringScoreValue.setText(""+CurrScore);
   JOptionPane.showMessageDialog(windowPlayGameLv1,"Congr
atulation,                Level                "+CurrLv+"
Accomplish","Message",JOptionPane.INFORMATION_MESSAGE);
          CurrLv=2;
          LoadMap(CurrLv);
          lblMScoringLevelValue.setText(""+CurrLv);
          lblMScoringStepValue.setText(""+StepLv2);
}}
```

Permainan level pertama di mulai dengan mengunci semua gembok yang ada pada level 1. Pada level pertama ini, diberikan langkah sebanyak 50 langkah. Setiap langkah yang di jalankan akan di kurangi dengan langkah awal. Jika pemain berhasil membuka gembok sebelum langkahnya habis maka permainan level pertama selesai.

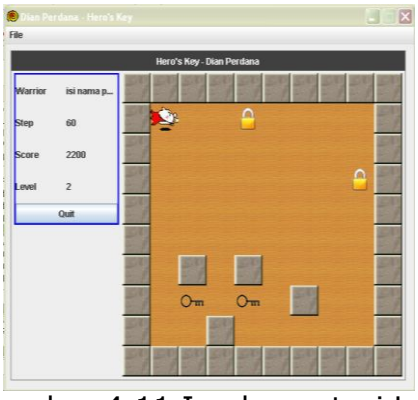
Gembok pada level pertama telah terbuka, permainan level pertamapun telah selesai, maka akan keluar *message box* seperti berikut ini :

Gambar 4.10 Gembok Terbuka Level 1

Setelah mengklik OK maka akan langsung menuju pada level selanjutnya.

f. Implementasi Level 2

Pada level kedua tingkat kesulitan untuk membuka gembok sudah mulai sedikit sulit. Pada level kedua ini, gembok berjumlah dua buah. Berikut ini adalah tampilan sebelum gembok terbuka:



Gambar 4.11 Implementasi Level 2

Source code dari implementasi jendela level kedua :

```
if(CurrLv==2)
{     LockLv2--;
```

```
if(LockLv2==0)
{      CurrStep=CurrStep+(60-StepLv2);
       CurrScore=CurrScore+(StepLv2*100);
       lblMScoringScoreValue.setText(""+CurrScore);
JOptionPane.showMessageDialog(windowPlayGameLv1,"Congr
atulation,                Level                "+CurrLv+"
Accomplish","Message",JOptionPane.INFORMATION_MESSAGE);
       CurrLv++;
       LoadMap(CurrLv);
       lblMScoringLevelValue.setText(""+CurrLv);
       lblMScoringStepValue.setText(""+StepLv3); }}
```

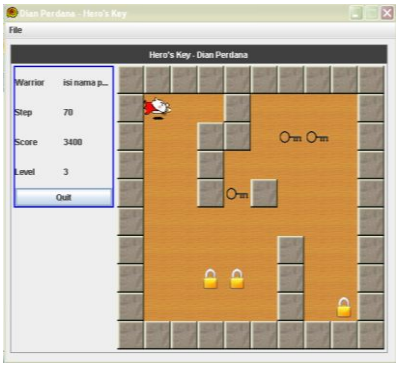
Apabila gembok yang terkunci telah terbuka maka telah menyelesaikan permainan level kedua, maka akan keluar *message box* seperti berikut ini :

Gambar 4.12 Gembok Terbuka Level Kedua

Setelah mengklik OK maka akan langsung menuju pada level selanjutnya.

g. Implementasi Level 3

Pada level ketiga, permainan akan semakin sulit. Dengan adanya rintangan dan jumlah gembok yang terkunci pun bertambah. Berikut adalah tampilannya :

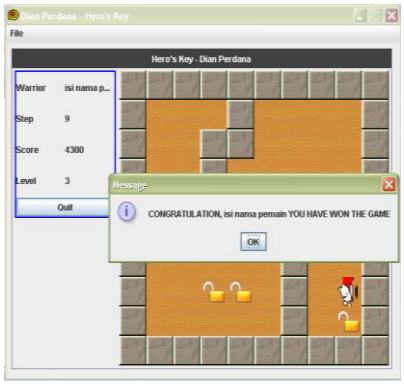


Gambar 4.13 Implementasi Level 3

Berikut adalah source code implementasi jendela level ketiga :

```
if(CurrLv==3)
{    LockLv3--;
     if(LockLv3==0)
     {      CurrStep=CurrStep+(70-StepLv3);
            CurrScore=CurrScore+(StepLv3*100);
            lblMScoringScoreValue.setText(""+CurrScore);
     //====WON WON
     JOptionPane.showMessageDialog(windowPlayGameLv1,"CONGR
ATULATION,     "+PlayedUser+"     YOU     HAVE     WON     THE
GAME","Message",JOptionPane.INFORMATION_MESSAGE);
     setEnabled(true);
```

Semua gembok telah terbuka dan *step* pun masih bersisa maka level terakhirpun selesai. Apabila telah menyelesaikan permainan level ketiga, maka akan keluar *message box* seperti berikut ini :

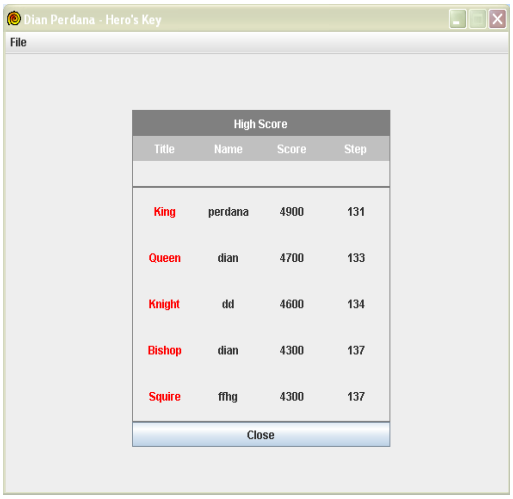


Gambar 4.14 Gembok Terbuka Level 3

Setelah menyelesaikan permainan maka klik OK dan akan langsung kembali ke menu awal.

h. Implementasi Jendela *Score*

Berikut ini adalah tampilan menu *high score*, di sini *king* sebagai peringkat pertama, *queen* sebagai peringkat kedua, *knight* sebagai peringkat ketiga, *bishop* sebagai peringkat keempat, dan *squire* sebagai peringkat ke lima.



Gambar 4.15 Implementasi Jendela *Score*