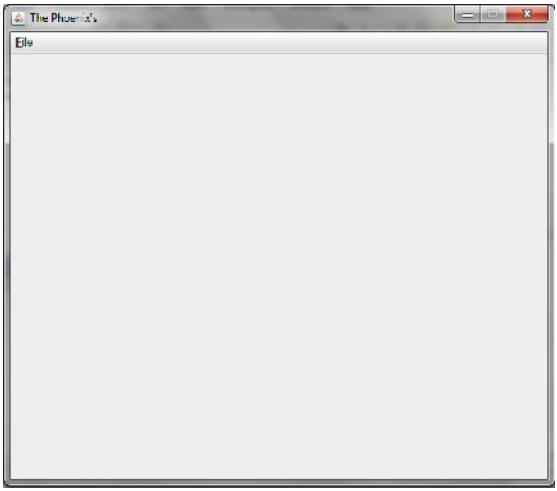
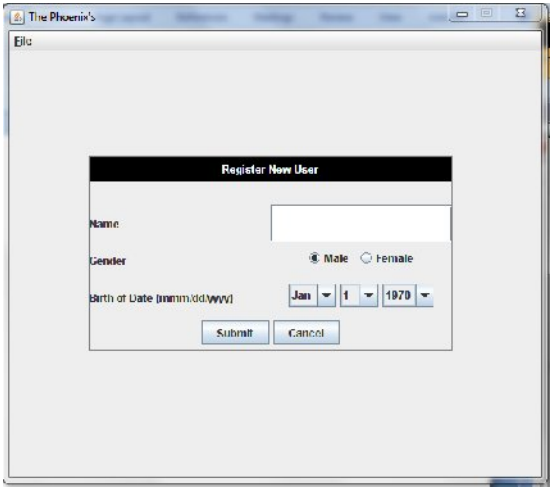
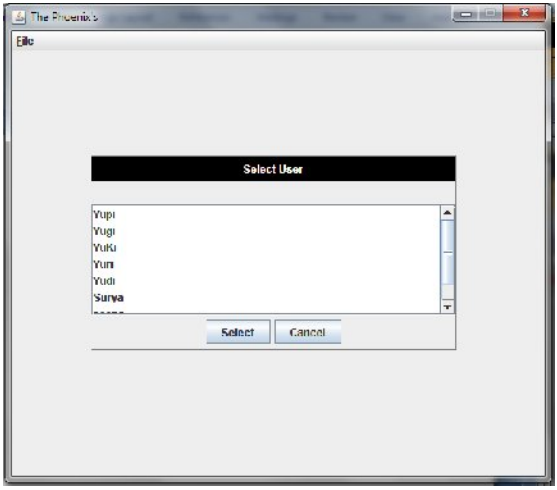
LAMPIRAN



Lampiran L1. Tampilan *Form* Utama

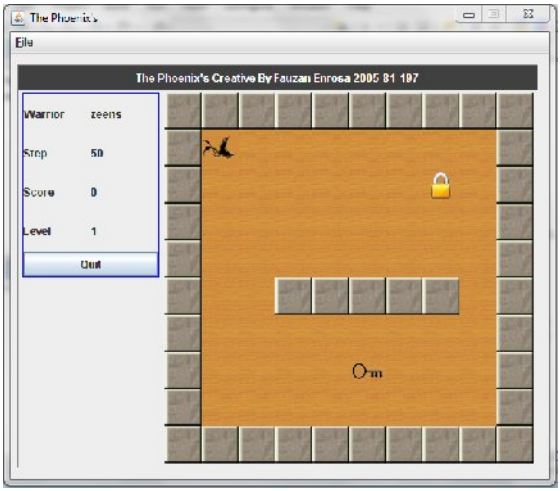


Lampiran L2. Tampilan *Form* Registrasi

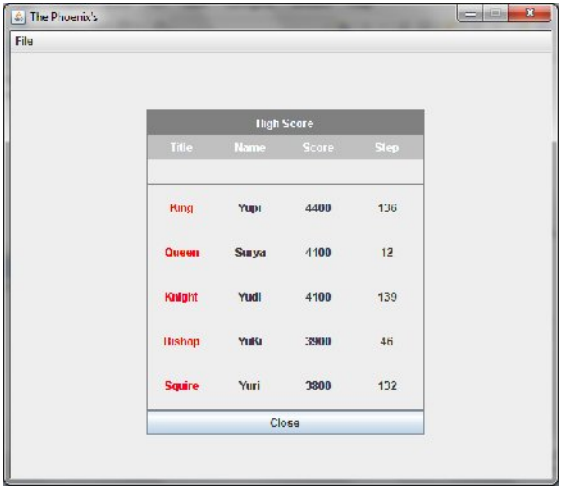


Lampiran L3. Tampilan *Form* Pemilihan *User*

Lampiran L4. Tampilan *Form* Sebelum Masuk Ke *Room*



Lampiran L5. Tampilan *Form Room Game* The Phoenix's



Lampiran L6. Tampilan *Form High Score*

Berikut ini adalah *source code program* dari aplikasi *game* The Phoenix's :

```
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.*;

class phoenix extends JFrame implements ActionListener, KeyListener
{
//pesen objek MENUBAR
        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        JMenuItem menuNewgame = new JMenuItem("New Game");
        JMenuItem menuScore = new JMenuItem("High Score");
        JMenuItem menuExit = new JMenuItem("Exit");
        JMenu menuRegis = new JMenu("Registration");
        JMenuBar menuBar1 = new JMenuBar();
//JMenu fileMenu1 = new JMenu("About");
        JMenuItem menuAbout = new JMenuItem("About Me");

// Pesen LABEL untuk JWindow HIGH SCORE
        JLabel lblHScore = new JLabel("High Score",JLabel.CENTER);
        JLabel lblBlank = new JLabel("      ");
        JLabel lblTitle = new JLabel("Title",JLabel.CENTER);
        JLabel lblName = new JLabel("Name",JLabel.CENTER);
```

```
        JLabel lblScore =  new JLabel("Score",JLabel.CENTER);
        JLabel lblStep = new JLabel("Step",JLabel.CENTER);
        JLabel lblKing = new JLabel("King",JLabel.CENTER);
        JLabel lblQueen = new JLabel("Queen",JLabel.CENTER);
        JLabel lblKnight = new JLabel("Knight",JLabel.CENTER);
        JLabel lblBishop = new JLabel("Bishop",JLabel.CENTER);
        JLabel lblSquire = new JLabel("Squire",JLabel.CENTER);
        JLabel lblStripSquireName = new JLabel("---",JLabel.CENTER);
        JLabel lblStripBishopName = new JLabel("---",JLabel.CENTER);
        JLabel lblStripKnightName = new JLabel("---",JLabel.CENTER);
        JLabel lblStripQueenName = new JLabel("---",JLabel.CENTER);
        JLabel lblStripKingName = new JLabel("---",JLabel.CENTER);

        JLabel lblStripSquireScore = new JLabel("---",JLabel.CENTER);
        JLabel lblStripBishopScore = new JLabel("---",JLabel.CENTER);
        JLabel lblStripKnightScore = new JLabel("---",JLabel.CENTER);
        JLabel lblStripQueenScore = new JLabel("---",JLabel.CENTER);
        JLabel lblStripKingScore = new JLabel("---",JLabel.CENTER);

        JLabel lblStripSquireStep = new JLabel("---",JLabel.CENTER);
        JLabel lblStripBishopStep = new JLabel("---",JLabel.CENTER);
        JLabel lblStripKnightStep = new JLabel("---",JLabel.CENTER);
        JLabel lblStripQueenStep = new JLabel("---",JLabel.CENTER);
        JLabel lblStripKingStep = new JLabel("---",JLabel.CENTER);
// LABEL tambahan untuk Frame NEW USER
        JLabel lblNUName = new JLabel("Name");
        JLabel lblGender = new JLabel("Gender");
        JLabel lblBOD = new JLabel("Birth of Date (mmm/dd/yyyy)");
JLabel lblRNUser =  new JLabel("Register New User",JLabel.CENTER);
```

```
// LABEL tambahan untuk Frame VIEW USER
        JLabel xx = new JLabel("View Registered User",JLabel.CENTER);
// LABEL tambahan untuk Frame VIEW USER
        JLabel lblSelectUser = new JLabel("Select User");
// pemesanan JLabel MENU SCORING
        JLabel lblMScoringWarrior = new JLabel("Warrior");
        JLabel lblMScoringStep = new JLabel("Step");
        JLabel lblMScoringScore = new JLabel("Score");
        JLabel lblMScoringLevel = new JLabel("Level");
        JLabel lblMScoringWarriorValue = new JLabel();
// menampung isi variabel
        JLabel lblMScoringStepValue = new JLabel();
        JLabel lblMScoringScoreValue = new JLabel();
        JLabel lblMScoringLevelValue = new JLabel();
//label menu start game
JLabel lblStartGameJdl = new JLabel("The Phoenix's Creative By Fauzan
Enrosa 2005-81-197",JLabel.CENTER);
// pesen Label windowPlay
        JLabel lIsiMapLv1[][] = new JLabel[10][10];

// pemesanan PANEL2
// Panel2 Window HIGH SCORE
        JPanel pNorth = new JPanel();
        JPanel pJudul =new JPanel();
        JPanel pCenter = new JPanel();
        JPanel pKet = new JPanel();
        JPanel pIsiKing = new JPanel();
        JPanel pIsiQueen = new JPanel();
        JPanel pIsiKnight = new JPanel();
```

```
        JPanel pIsiBishop = new JPanel();
        JPanel pIsiSquire = new JPanel();
//PANEL2 Frame NEW USER
        JPanel pWindowNUser = new JPanel();
        JPanel pNUserNorth = new JPanel();
                JPanel pRNUser = new JPanel();
        JPanel pNUserCenter = new JPanel();
                JPanel pNUserCenterGender = new JPanel();
                JPanel pNUserCenterBOD = new JPanel();
        JPanel pNUserSouth = new JPanel();
// PANEL2 Frame VIEW REGISTERED USER
        JPanel pView = new JPanel();
        JPanel pViewNorth = new JPanel();
// pemesanan panel2 SELECT USER
        JPanel pWindowSelectUsr = new JPanel();
        JPanel pSelectUsrNorth = new JPanel();
        JPanel pSelectUser = new JPanel();
        JPanel pSelectUsrCenter = new JPanel();
        JPanel pSelectUsrSouth = new JPanel();
// panel2 window start game
        JPanel pStartGame = new JPanel();
        JPanel pStartGameNorth = new JPanel();
        JPanel pStartGameJudul =new JPanel();
// Panel2 Menu Scoring
        JPanel pMenuScoring = new JPanel();
        JPanel pMenuScoringCenter = new JPanel();
        JPanel pMenuScoringSouth = new JPanel();
// panel window play
        JPanel pWindowPlayGameLv1 = new JPanel();
```

```
        JPanel pPlayGamePapanLv1 = new JPanel();

// pesen objek BUTTON
        JButton btnClose = new JButton("Close"); // -> close yang di window
score
        JButton btnNUserSubmit = new JButton("Submit");
        JButton btnNUserCancel = new JButton("Cancel");
//  BUTTON  frame ViewUsers
        JButton btnfirst = new JButton("<<");
        JButton btnprev = new JButton("<");
        JButton btnnext = new JButton(">");
        JButton btnlast = new JButton(">>");
        JButton btnViewCancel = new JButton("Cancel");
        JButton btnViewUpdate = new JButton("Update");
//  BUTTON  frame select Users
        JButton btnSUserSelect = new JButton("Select");
        JButton btnSUserCancel = new JButton("Cancel");
//button menu scoring
        JButton btnMScoringQuit = new JButton("Quit");

// pemesanan TEXTFIELD
        JTextField txtNama = new JTextField();
        JTextField VtxtNama;

//pemesanan Jlist
        JList listUser = new JList();

// pemesanan RADIO BUTTON
        JRadioButton optMale = new JRadioButton("Male");
```

```
        JRadioButton optFemale = new JRadioButton("Female");
        JRadioButton VoptMale = new JRadioButton("Male");
        JRadioButton VoptFemale = new JRadioButton("Female");

//pemesanan COMBOBOX
        JComboBox comboDay = new JComboBox();
        JComboBox comboMonth = new JComboBox();
        JComboBox comboYear = new JComboBox();
        JComboBox VcomboDay = new JComboBox();
        JComboBox VcomboMonth = new JComboBox();
        JComboBox VcomboYear = new JComboBox();

//pemesanan String untuk bulan
        String MONTH[] =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Nov","Oct","Dec"};
        String PlayedUser ="";

        int current = 0; // digunakan pada proses ViewUser, sebagai penanda
posisi Current


// window start game
        int mapData[][] = new int[10][10]; // untuk menyimpan data map dari
map.txt
        int mapData1[][] = new int[10][10];
        int mapData2[][] = new int[10][10];
        int mapData3[][] = new int[10][10];

        int heroPosX =1;
```

```
            int heroPosY =1;
            int z=0;

            int CurrLv=1;
            int LockLv1 = 1;
            int LockLv2 = 2;
            int LockLv3 = 3;

            int StepLv1 = 50;
            int StepLv2 = 60;
            int StepLv3 = 70;

            int CurrScore = 0;
            int CurrStep = 0;

//panggil image iconnya
            ImageIcon imgBatu = new ImageIcon("Images/batu1.gif");
            ImageIcon imgKunci = new ImageIcon("Images/batu2.gif");
            ImageIcon imgGembokLock = new ImageIcon("Images/cewe.GIF");
            ImageIcon imgGembokOpen = new
ImageIcon("Images/mariage.GIF");
            ImageIcon imgTanah = new ImageIcon("Images/tanah.GIF");
            ImageIcon imgHeroUP = new ImageIcon("Images/hero1.GIF");
            ImageIcon imgHeroDOWN = new ImageIcon("Images/hero2.GIF");
            ImageIcon imgHeroLEFT = new ImageIcon("Images/hero3.GIF");
            ImageIcon imgHeroRIGHT = new ImageIcon("Images/hero4.GIF");


// pemesanan vektor variabel global untuk proses ViewUser
```

```
        Vector peoplesTemp = new Vector();    // -> data2 user akan disimpan
disini setelah diLoad
        Vector vectorSelectUser = new Vector();

// pemesanan objek untuk menu bar
        JMenuItem menuRegisNew = new JMenuItem("New User");
        JMenuItem menuRegisView = new JMenuItem("View User");

// pesen objek JWindow & JFrame
        JWindow windowScore = new JWindow();
        JFrame winNUser = new JFrame();
        JFrame windowViews = new JFrame();
        JWindow windowSelectUsr = new JWindow();
        JWindow windowStartGame = new JWindow();
        JWindow windowMenuScoring = new JWindow();
        JFrame windowPlayGameLv1 = new JFrame();

// pesen Objek Container
        Container c = getContentPane();
        Container cScore = windowScore.getContentPane(); // container untuk
window Score
        Container cNUser = winNUser.getContentPane();          //
container untuk frame new user
        Container con = windowViews.getContentPane();          //
container untuk frame view user
        Container cSelectUsr = windowSelectUsr.getContentPane();
// container untuk frame select user
        Container cWindowStartGame = windowStartGame.getContentPane();
// container untuk frame window start game
```

```
        Container cMenuScoring = windowMenuScoring.getContentPane();
//container untuk frame menu scoring
        Container cPlayGameLv1 = windowPlayGameLv1.getContentPane();
//container untuk frame playgame (field permainan)

//---------- KONSTRUKTOR -------------
        phoenix()
        {
                setTitle("The Phoenix's");
                setIconImage( (new
ImageIcon("Images/ani780.ico")).getImage() );
                setBounds(          90, 40, 590, 510);

        // konfigurasi layout utama
                c.setLayout(null);

//---------------------- KONFIGURASI layout untuk WINDOW SCORE  -----------
// konfigurasi posisi window score
                windowScore.setBounds(      240, 150, 250, 250);
//konfigurasi panel
//panelMain.setLayout(new BorderLayout());
                pJudul.add(lblHScore);
                pNorth.setLayout(new GridLayout(3,1));
                pNorth.add(pJudul);
                pNorth.add(lblBlank);
                pNorth.add(pKet);

// isi panel keterangan
                pKet.setLayout(new GridLayout(1,4));
```

```
        pKet.add(lblTitle);
        pKet.add(lblName);
        pKet.add(lblScore);
        pKet.add(lblStep);



// isi panel Center ----------------------
        pCenter.setLayout(new GridLayout(5,1));
        pCenter.add(pIsiKing);
        pCenter.add(pIsiQueen);
        pCenter.add(pIsiKnight);
        pCenter.add(pIsiBishop);
        pCenter.add(pIsiSquire);

// isi panel King
        pIsiKing.setLayout(new GridLayout(1,4));
        pIsiKing.add(lblKing);
        pIsiKing.add(lblStripKingName);
        pIsiKing.add(lblStripKingScore);
        pIsiKing.add(lblStripKingStep);

// isi panel Queen
        pIsiQueen.setLayout(new GridLayout(1,4));
        pIsiQueen.add(lblQueen);
        pIsiQueen.add(lblStripQueenName);
        pIsiQueen.add(lblStripQueenScore);
        pIsiQueen.add(lblStripQueenStep);
```

```
// isi panel Knight
                    pIsiKnight.setLayout(new GridLayout(1,4));
                    pIsiKnight.add(lblKnight);
                    pIsiKnight.add(lblStripKnightName);
                    pIsiKnight.add(lblStripKnightScore);
                    pIsiKnight.add(lblStripKnightStep);

// isi panel Bishop
                    pIsiBishop.setLayout(new GridLayout(1,4));
                    pIsiBishop.add(lblBishop);
                    pIsiBishop.add(lblStripBishopName);
                    pIsiBishop.add(lblStripBishopScore);
                    pIsiBishop.add(lblStripBishopStep);

// isi panel Squire
                    pIsiSquire.setLayout(new GridLayout(1,4));
                    pIsiSquire.add(lblSquire);
                    pIsiSquire.add(lblStripSquireName);
                    pIsiSquire.add(lblStripSquireScore);
                    pIsiSquire.add(lblStripSquireStep);



// pewarnaan text dan backgrounds
                    lblHScore.setForeground(Color.WHITE);
                    lblTitle.setForeground(Color.WHITE);
                    lblName.setForeground(Color.WHITE);
                    lblScore.setForeground(Color.WHITE);
                    lblStep.setForeground(Color.WHITE);
```

```
                    lblKing.setForeground(Color.RED);
                    lblQueen.setForeground(Color.RED);
                    lblKnight.setForeground(Color.RED);
                    lblBishop.setForeground(Color.RED);
                    lblSquire.setForeground(Color.RED);

                    cScore.add(pNorth,"North");
                    cScore.add(pCenter,"Center");
                    cScore.add(btnClose,"South");
                    pJudul.setBackground(Color.GRAY);
                    pKet.setBackground(Color.LIGHT_GRAY);

          pNorth.setBorder(BorderFactory.createLineBorder(Color.GRAY));
          pCenter.setBorder(BorderFactory.createLineBorder(Color.GRAY));
                    windowScore.setSize(300,350);

//---------- END OF - Konfigurasi layout untuk JWindow Score - END ----------

//----------------------KONFIGURASI FRAME NEW USER-------------------------

// konfigurasi posisi frame NEW USER
                    winNUser.setUndecorated(true);
                    winNUser.setBounds(180, 200, 200, 200);
//txtNama.setEnabled(true);
//cNUser.setEnabled(true);
                    cNUser.setLayout(new BorderLayout());

//konfig untuk radio button agar menjadi 1 group
                    ButtonGroup group = new ButtonGroup();
```

```
                    group.add(optMale);
                    group.add(optFemale);

//input isi BOD
//comboDay.removeAll();

                    for(int i=1;i<=31;i++) { comboDay.addItem(i); }
                    for(int i=1970;i<2000;i++) { comboYear.addItem(i); }
                    for(int i=0;i<12;i++) { comboMonth.addItem(MONTH[i]);
}
                    comboYear.addActionListener(this);
                    comboMonth.addActionListener(this);

//konfigurasi panel2

                    pWindowNUser.setLayout(new BorderLayout());
                    pWindowNUser.add(pNUserNorth,"North");
                    pWindowNUser.add(pNUserCenter,"Center");
                    pWindowNUser.add(pNUserSouth,"South");

//----panel NORTH
                    pRNUser.add(lblRNUser);
                    pNUserNorth.setLayout(new GridLayout(2,1));
                    pNUserNorth.add(pRNUser);
                    pNUserNorth.add(lblBlank);
```

```
//----panel CENTER
                    pNUserCenter.setLayout(new GridLayout(3,2));
//---nama
                    pNUserCenter.add(lblNUName);
                    pNUserCenter.add(txtNama);
//---gender
                    pNUserCenter.add(lblGender);
                    pNUserCenter.add(pNUserCenterGender);

                    pNUserCenterGender.setLayout(new FlowLayout());
                    pNUserCenterGender.add(optMale);
                    pNUserCenterGender.add(optFemale);
//---BOD
                    pNUserCenter.add(lblBOD);
                    pNUserCenter.add(pNUserCenterBOD);

                    pNUserCenterBOD.setLayout(new FlowLayout());
                    pNUserCenterBOD.add(comboMonth);
                    pNUserCenterBOD.add(comboDay);
                    pNUserCenterBOD.add(comboYear);

//----panel SOUTH
                    pNUserSouth.setLayout(new FlowLayout());
                    pNUserSouth.add(btnNUserSubmit);
                    pNUserSouth.add(btnNUserCancel);

// pesan untuk action listener
                    btnNUserSubmit.addActionListener(this);
                    btnNUserCancel.addActionListener(this);
```

```
// pewarnaan text dan backgrounds & border
                    lblRNUser.setForeground(Color.WHITE);
                    pRNUser.setBackground(Color.BLACK);

pWindowNUser.setBorder(BorderFactory.createLineBorder(Color.GRAY));
                    cNUser.add(pWindowNUser);
                    winNUser.setSize(400,210);

//----------END----------KONFIGURASI FRAME NEW USER--------- END

//--------KONFIGURASI FRAME VIEW REGISTERED USER------

                    windowViews.setUndecorated(true);
                    con.setLayout(new BorderLayout());

                    ButtonGroup Vgroup = new ButtonGroup();
                    group.add(VoptMale);
                    group.add(VoptFemale);

                    for(int i=1;i<=31;i++) { VcomboDay.addItem(i); }
                    for(int i=1970;i<2000;i++) { VcomboYear.addItem(i); }
                    for(int i=0;i<12;i++) { VcomboMonth.addItem(MONTH[i]);
}

//----------------------
                    JPanel panel1 = new JPanel(new GridLayout(3,2));
                    panel1.add(new JLabel("Name"));
                    panel1.add(VtxtNama = new JTextField());
```

```
                    panel1.add(new JLabel("Gender"));
                    JPanel pvGender = new JPanel(new FlowLayout());
                    panel1.add(pvGender);
                            pvGender.add(VoptMale);
                            pvGender.add(VoptFemale);

                    JPanel pvBOD = new JPanel(new FlowLayout());
                    panel1.add(new JLabel("Birth Of Date (mmm/dd/yyyy)"));
                    panel1.add(pvBOD);

                    pvBOD.add(VcomboMonth);
                    pvBOD.add(VcomboDay);
                    pvBOD.add(VcomboYear);

                    VtxtNama.setEnabled(false);
//ph.setEnabled(false);
//email.setEnabled(false);
//----------------
                    JPanel panel2 = new JPanel(new FlowLayout());
                    panel2.add(btnfirst);
                    panel2.add(btnprev);
                    panel2.add(btnViewUpdate);
                    panel2.add(btnViewCancel);
                    panel2.add(btnnext);
                    panel2.add(btnlast);

                    btnfirst.addActionListener(this);
                    btnprev.addActionListener(this);
                    btnnext.addActionListener(this);
```

```
                btnlast.addActionListener(this);
                btnViewUpdate.addActionListener(this);
                btnViewCancel.addActionListener(this);

                xx.setForeground(Color.WHITE);
                pViewNorth.add(xx);

                pView.setLayout(new BorderLayout());
                pView.add(panel1,BorderLayout.CENTER);
                pView.add(panel2,BorderLayout.SOUTH);
                pView.add(pViewNorth,BorderLayout.NORTH);

        pView.setBorder(BorderFactory.createLineBorder(Color.GRAY));

                pViewNorth.setBackground(Color.GRAY);
                con.add(pView,BorderLayout.CENTER);
                windowViews.setBounds(180,200,400,200);

//-END---END--KONFIGURASI FRAME VIEW REGISTERED USER--END-

//----------------------KONFIGURASI FRAME SELECT USER-----------------------

//windowSelectUsr.setUndecorated(true);
                windowSelectUsr.setBounds( 180, 200, 200, 200);
                cSelectUsr.setLayout(new BorderLayout());

//konfigurasi panel2
                pWindowSelectUsr.setLayout(new BorderLayout());
                pWindowSelectUsr.add(pSelectUsrNorth,"North");
```

```
                pWindowSelectUsr.add(pSelectUsrCenter,"Center");
                pWindowSelectUsr.add(pSelectUsrSouth,"South");

//----panel NORTH
                pSelectUser.add(lblSelectUser);
                pSelectUsrNorth.setLayout(new GridLayout(2,1));
                pSelectUsrNorth.add(pSelectUser);
                pSelectUsrNorth.add(lblBlank);

//----panel CENTER
                pSelectUsrCenter.setLayout(new BorderLayout());
                pSelectUsrCenter.add(new JScrollPane(listUser)); // List
Box
//----panel SOUTH
                pSelectUsrSouth.setLayout(new FlowLayout());
                pSelectUsrSouth.add(btnSUserSelect);
                pSelectUsrSouth.add(btnSUserCancel);

// pesan untuk action listener
                btnSUserSelect.addActionListener(this);
                btnSUserCancel.addActionListener(this);

// pewarnaan text dan backgrounds & border
                lblSelectUser.setForeground(Color.WHITE);
                pSelectUser.setBackground(Color.BLACK);

pWindowSelectUsr.setBorder(BorderFactory.createLineBorder(Color.GRAY));
                cSelectUsr.add(pWindowSelectUsr);
```

```
                    listUser.setListData(vectorSelectUser);
                    windowSelectUsr.setSize(400,210);

//try{loadFile2();}catch(Exception r) {}
/*
                    vectorSelectUser.removeAllElements();
                    for(int i=0;i<peoplesTemp.size();i++)
                    {
                            String people = (String)
peoplesTemp.elementAt(i);
                            String split[] = people.split(";");
                            vectorSelectUser.add(split[0]);
                    }

*/
//----END----KONFIGURASI FRAME SELECT USER--------END------END---

//--- WINDOW START GAME

//----WINDOW START GAME
//                  windowStartGame.setUndecorated(true);
                    windowStartGame.setBounds(100, 100, 565, 435);

//konfigurasi panel
                    pStartGame.setLayout(new BorderLayout());
                    pStartGame.add(pStartGameNorth,"North");
                    pStartGameJudul.add(lblStartGameJdl);
                    pStartGameNorth.setLayout(new GridLayout(1,1));
                    pStartGameNorth.add(pStartGameJudul);
```

```
// pewarnaan text dan backgrounds
                    lblStartGameJdl.setForeground(Color.WHITE);
                    pStartGameJudul.setBackground(Color.DARK_GRAY);
                    cWindowStartGame.add(pStartGame);

pStartGame.setBorder(BorderFactory.createLineBorder(Color.GRAY));

                    windowStartGame.setEnabled(false);
//windowStartGame.setSize(580,600);

//------END WINDOW START GAME

//------MENU SCORING
//                  windowMenuScoring.setUndecorated(true);
                    windowMenuScoring.setBounds(105, 130, 150, 200);

                    lblMScoringStepValue.setText(""+StepLv1);
                    lblMScoringScoreValue.setText(""+CurrScore);
                    lblMScoringLevelValue.setText(""+CurrLv);

                    pMenuScoringCenter.setLayout(new GridLayout(4,2));
                    pMenuScoringCenter.add(lblMScoringWarrior);
                    pMenuScoringCenter.add(lblMScoringWarriorValue);
                    pMenuScoringCenter.add(lblMScoringStep);
                    pMenuScoringCenter.add(lblMScoringStepValue);
                    pMenuScoringCenter.add(lblMScoringScore);
                    pMenuScoringCenter.add(lblMScoringScoreValue);
                    pMenuScoringCenter.add(lblMScoringLevel);
```

```
                    pMenuScoringCenter.add(lblMScoringLevelValue);
                    pMenuScoringSouth.setLayout(new BorderLayout());
                    pMenuScoringSouth.add(btnMScoringQuit);
                    pMenuScoring.setLayout(new BorderLayout());
                    pMenuScoring.add(pMenuScoringCenter,"Center");
                    pMenuScoring.add(pMenuScoringSouth,"South");

                    cMenuScoring.add(pMenuScoring);

                    //windowMenuScoring.setVisible(true);

// BORDER

pMenuScoring.setBorder(BorderFactory.createLineBorder(Color.BLUE,2));

//pesen action listener
                    btnMScoringQuit.addActionListener(this);

//------END MENU SCORING

//------WINDOW PLAY
// konfigurasi posisi frame NEW USER
                    windowPlayGameLv1.setUndecorated(true);
                    windowPlayGameLv1.setBounds( 260, 130, 400, 400);
                    cPlayGameLv1.setLayout(new BorderLayout());
//konfigurasi panel2
                    pWindowPlayGameLv1.setLayout(new BorderLayout());
                    pWindowPlayGameLv1.add(pPlayGamePapanLv1);
```

```
//----panel CENTER
                    pPlayGamePapanLv1.setLayout(new GridLayout(10,10));
                    LoadFileMap();

//LoadMap(CurrLv);
                    cPlayGameLv1.add(pWindowPlayGameLv1);

//windowPlayGameLv1.setSize(400,400);
//windowPlayGameLv1.setVisible(true);

                    windowPlayGameLv1.addKeyListener(this);

//-------END WINDOW PLAY

//-------END WINDOW START GAME

//buat menubar
                    setJMenuBar(menuBar);
//setJMenuBar(menuBar1);
                    menuBar.add(fileMenu);
//menuBar1.add(fileMenu1);

                    fileMenu.add(menuNewgame);
                    fileMenu.add(menuRegis);
                    fileMenu.add(new JSeparator());
                    fileMenu.add(menuScore);
                    fileMenu.add(menuAbout);
                    fileMenu.add(menuExit);
```

```
                menuRegis.add(menuRegisNew);
                menuRegis.add(menuRegisView);

// buat Shortcut menu
                fileMenu.setMnemonic('F');


menuNewgame.setAccelerator(KeyStroke.getKeyStroke('N',CTRL_DOWN_M
ASK));
menuScore.setAccelerator(KeyStroke.getKeyStroke('H',CTRL_DOWN_MASK)
);
menuExit.setAccelerator(KeyStroke.getKeyStroke('X',CTRL_DOWN_MASK));
menuAbout.setAccelerator(KeyStroke.getKeyStroke('A',CTRL_DOWN_MASK
));
menuRegisNew.setAccelerator(KeyStroke.getKeyStroke('W',CTRL_DOWN_M
ASK));
menuRegisView.setAccelerator(KeyStroke.getKeyStroke('U',CTRL_DOWN_M
ASK));

//bwat icon menubar
                menuNewgame.setIcon(new ImageIcon("Images/new.gif"));
                menuScore.setIcon(new ImageIcon("Images/score.png"));
                menuExit.setIcon(new ImageIcon("Images/exit.png"));
                menuRegisNew.setIcon(new ImageIcon("Images/user.gif"));
                menuRegisView.setIcon(new
ImageIcon("Images/view.png"));

// pesen action listener
                menuNewgame.addActionListener(this);
```

```
                menuScore.addActionListener(this);
                menuExit.addActionListener(this);
                menuRegisNew.addActionListener(this);
                menuRegisView.addActionListener(this);
                btnClose.addActionListener(this); //-> terdapat pada window
SCORE

//setSize(650,600);
                setResizable(false);
                show();
                setVisible(true);
                setResizable(false);
                setDefaultCloseOperation(3);
        }

//-----END KONSTRUKTOR;

        public static void main(String[]args)
        {
                new phoenix();
        }
        public void actionPerformed(ActionEvent e)
        {
                Object source = e.getSource();
                if(source==menuNewgame)
                {
                        //try{loadFile2();}catch(Exception r) {}
                        //JOptionPane.showMessageDialog(null, "There
Are No User", "Warning", JOptionPane.ERROR_MESSAGE);
```

```
                        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();
                        }
                }
                else if(source==menuScore)
                {
//JOptionPane.showMessageDialog(null,"ini Menu Score");
                        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();
                        }
                        else if(source==menuRegisNew)
                        {
            // reset field new user
                                txtNama.setText("");
                                optMale.setSelected(true);
                                comboDay.removeAllItems();
                                for(int i=1;i<=31;i++) { comboDay.addItem(i); }
                                comboMonth.setSelectedItem("Jan");
                                comboDay.setSelectedItem(1);
                                comboYear.setSelectedItem(1970);

            //JOptionPane.showMessageDialog(null,"ini Menu New Player");
                                btnNUserSubmit.setVisible(true);
                                btnNUserCancel.setVisible(true);
                                setEnabled(false);
                                winNUser.setVisible(true);
                                winNUser.show();

                        }
                        else if(source==menuRegisView)
```

```
                {
        //JOptionPane.showMessageDialog(null,"ini Menu View Player");
                        if(initialize()==1)
                        {
                        //baca dari peopletemp

                                btnfirst.setVisible(true);
                                btnprev.setVisible(true);
                                btnnext.setVisible(true);
                                btnlast.setVisible(true);
                                btnViewCancel.setVisible(true);
                                btnViewUpdate.setVisible(true);
                                setEnabled(false);
                                windowViews.setVisible(true);
                                windowViews.show();
                        }
                }
                else if(source==menuExit)
                {
                        dispose();
                        System.exit(0);
                }
                else if(source==btnClose)
                {
                        windowScore.dispose();
                        setEnabled(true);
                }
//-----------------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;
                        }
String tempNamaUser=txtNama.getText();  // menyimpan nama user
String msg = tempNamaUser +"\n";      // menampung message autosifikasi

                // validasi gender
                        String tempGenderUser;
                // menyimpan jenis gender
                        if(optMale.isSelected())
                        {
                                tempGenderUser="Male";
                                msg = msg + tempGenderUser +"\n";
                        }
                        else if(optFemale.isSelected()) {
                                tempGenderUser="Female";
                                msg = msg + tempGenderUser +"\n";
                        }
                        else
                        {
```

```
JOptionPane.showMessageDialog(null,"Please Chose Your
Gender","Information",JOptionPane.INFORMATION_MESSAGE);
return;
}

String tempMonth = comboMonth.getSelectedItem().toString();
// menyimpan Birth of Date - Bulan
String tempDay = comboDay.getSelectedItem().toString();
// menyimpan Birth of Date - Hari
String tempYear = comboYear.getSelectedItem().toString();
// menyimpan Birth of Date - Tahun

msg = msg + tempMonth +" "+ tempDay +" "+ tempYear +"\n\n";

if(JOptionPane.showConfirmDialog(null, msg+"Are You Sure With These Data
?", "Information",
JOptionPane.OK_CANCEL_OPTION,JOptionPane.QUESTION_MESSAGE)=
=0)
                                        {
// -----------------BACA DATA User di "user.sav"----------------------

                                                Vector peoples = new Vector();
                                                try
                                                {

        peoplesTemp.removeAllElements(); //tes
                                                        BufferedReader buf = new
BufferedReader(new FileReader("user.sav"));
                                                        String line = null;
```

```
            while((line=buf.readLine())!=null) {
            peoplesTemp.add(line);
                                                    }
                                                    buf.close();
                                                }catch(Exception ex) {  }

// ------Penyimpanan DATA USER -------SAVE OTOMATIS ke "user.sav"------

                                                try
                                                {
            PrintWriter p2 = new PrintWriter(new FileWriter("user.sav"));


// menulis kembali data user sebelumnya
            for(int i=0;i<peoplesTemp.size();i++)
                                                    {
            String people2 = (String) peoplesTemp.elementAt(i);
            String split2[] = people2.split(";");

            p2.print(split2[0]+";");
            p2.print(split2[1]+";");
            p2.print(split2[2]+";");
            p2.print(split2[3]+";");
            p2.print(split2[4]);
            p2.println();
            }
// menambah data yang baru
            p2.print(tempNamaUser+";");
```

```
            p2.print(tempGenderUser+";");
            p2.print(tempMonth+";");
            p2.print(tempDay+";");
            p2.print(tempYear);
            p2.println();
            p2.close();
            //loadFile();

            peoplesTemp.add(tempNamaUser+";"+tempGenderUser+";"+tempMo
nth+";"+tempDay+";"+tempYear);

            } catch(Exception ex) { }

// -END Penyimpanan DATA USER --END--SAVE OTOMATIS ke "user.sav"-
// informasi message

            JOptionPane.showMessageDialog(null,"Registration
Complete","Success",JOptionPane.INFORMATION_MESSAGE);
                                    winNUser.dispose();
                                    setEnabled(true);
                                    show();
                        }
            }
            else if(source==btnNUserCancel)
            {
                        winNUser.dispose();
                        setEnabled(true);
                        show();
```

```
                }

                //**** KABISAT
                /*
                        removeAllItems()
                        addItem(Object anObject)
                        getSelectedItem()
                        comboDay.removeAllItems();
                        for(int i=1;i<=31;i++) { comboDay.addItem(i); }
                        String MONTH[] =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Nov","Oct","Dec"};
                */
                else if(source==comboYear)
                {
                        int selectedYear =
Integer.parseInt(comboYear.getSelectedItem().toString());
                        String selectedMonth =
comboMonth.getSelectedItem().toString();
//System.out.println(selectedYear);
                        if(selectedMonth=="Feb")
                        {
                                comboDay.removeAllItems();
                        for(int i=1;i<=28;i++) { comboDay.addItem(i); }
                                if((selectedYear%4)==0)
                                {
                                        comboDay.removeAllItems();
                                        for(int i=1;i<=29;i++) {
comboDay.addItem(i); }
                                }
```

```
                    }
                }
                else if(source==comboMonth)
                {
String selectedMonth = comboMonth.getSelectedItem().toString();
                    int selectedYear =
Integer.parseInt(comboYear.getSelectedItem().toString());

//System.out.println(selectedMonth);
                    if(selectedMonth=="Apr" || selectedMonth=="Jun"
|| selectedMonth=="Sep" || selectedMonth=="Nov")
                    {
                        comboDay.removeAllItems();
                    for(int i=1;i<=30;i++) { comboDay.addItem(i); }
                    }
                    else if(selectedMonth=="Jan" ||
selectedMonth=="Mar" || selectedMonth=="May" || selectedMonth=="Jul"
||selectedMonth=="Aug" || selectedMonth=="Oct" || selectedMonth=="Dec")
                    {
                        comboDay.removeAllItems();
                    for(int i=1;i<=31;i++) { comboDay.addItem(i); }
                    }
                    else if(selectedMonth=="Feb")
                    {
                        comboDay.removeAllItems();
                    for(int i=1;i<=28;i++) { comboDay.addItem(i); }
                        if((selectedYear%4)==0)
                        {
                            comboDay.removeAllItems();
```

```
                        for(int i=1;i<=29;i++) { comboDay.addItem(i); }
                                }
                        }
                }

//*****Untuk Menu ViewUser +Update
                else if(source==VcomboYear)
                {
                        int selectedYear =
Integer.parseInt(VcomboYear.getSelectedItem().toString());
                        String selectedMonth =
VcomboMonth.getSelectedItem().toString();
        //System.out.println(selectedYear);
                        if(selectedMonth=="Feb")
                        {
                                VcomboDay.removeAllItems();
                        for(int i=1;i<=28;i++) { VcomboDay.addItem(i); }
                                if((selectedYear%4)==0)
                                {
                        VcomboDay.removeAllItems();
                        for(int i=1;i<=29;i++) { VcomboDay.addItem(i); }
                                }
                        }
                }
                else if(source==VcomboMonth)
                {
                String selectedMonth =
VcomboMonth.getSelectedItem().toString();
```

```
                                int selectedYear =
Integer.parseInt(VcomboYear.getSelectedItem().toString());

        //System.out.println(selectedMonth);
                                if(selectedMonth=="Apr" || selectedMonth=="Jun"
|| selectedMonth=="Sep" || selectedMonth=="Nov")
                                {
                                        VcomboDay.removeAllItems();
                                for(int i=1;i<=30;i++) { VcomboDay.addItem(i); }
                                }
                                else if(selectedMonth=="Jan" ||
selectedMonth=="Mar" || selectedMonth=="May" || selectedMonth=="Jul"
||selectedMonth=="Aug" || selectedMonth=="Oct" || selectedMonth=="Dec")
                                {
                                        VcomboDay.removeAllItems();
                                for(int i=1;i<=31;i++) { VcomboDay.addItem(i); }
                                }
                                else if(selectedMonth=="Feb")
                                {
                                        VcomboDay.removeAllItems();
                                for(int i=1;i<=28;i++) { VcomboDay.addItem(i); }
                                        if((selectedYear%4)==0)
                                        {

        VcomboDay.removeAllItems();
                                for(int i=1;i<=29;i++) { VcomboDay.addItem(i); }
                                        }
                                }
                        }
```

```
//******  END KABISAT
//-------------END----------- Listener NewUSER
//-----------------------------LISTENER FRAME VIEW --------------------------------
                        else if(source == btnfirst)
                        {
                                current = 0;
                                showPeople(current);
                        }
                        else if(source == btnprev)
                        {
                                current -= 1;
                                if(current<0) { current = 0; }
                                showPeople(current);
                        }
                        else if(source == btnnext)
                        {
                                current += 1;
                                if(current==peoplesTemp.size()) { current -=1; }
                                showPeople(current);
                        }
                        else if(source == btnlast)
                        {
                                current = peoplesTemp.size()-1;
                                showPeople(current);
                        }
                        else if(source == btnViewUpdate)
                        {
//update -------------------------------
```

```
// simpan data perubahan ke string

        String tempNamaUser=VtxtNama.getText();  // menyimpan nama user
        String tempGenderUser="";
// menyimpan jenis gender
                        if(VoptMale.isSelected())
                        {
                                tempGenderUser="Male";
                        }
                        else if(VoptFemale.isSelected())
                        {
                                tempGenderUser="Female";
                        }
        String tempMonth = VcomboMonth.getSelectedItem().toString();
// menyimpan Birth of Date - Bulan
        String tempDay = VcomboDay.getSelectedItem().toString();
// menyimpan Birth of Date - Hari
        String tempYear = VcomboYear.getSelectedItem().toString();
// menyimpan Birth of Date - Tahun
        // lakukan update
                                try
                                {
        PrintWriter p = new PrintWriter(new FileWriter("user.sav"));

// menulis kembali data user sebelumnya
                        for(int i=0;i<peoplesTemp.size();i++)
                                        {
                                                if(i!=current)
                                                {
```

```
                String people = (String) peoplesTemp.elementAt(i);
                                                        p.print(people);
                                                        }

// current adalah index dari data vektor yang ingin dirubah
                                                        else if(i==current)
                                                        {

        p.print(tempNamaUser+";");
        p.print(tempGenderUser+";");
        p.print(tempMonth+";");
        p.print(tempDay+";");
        p.print(tempYear);

                                                        String newData =
tempNamaUser+";"+tempGenderUser+";"+tempMonth+";"+tempDay+";"+temp
Year;
        peoplesTemp.remove(i);
        peoplesTemp.insertElementAt(newData, i);
                                                        }
                p.println();
                                                }
                p.close();

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


                                        } catch(Exception ex)
```

```
                                        {
                                        }
                        }
                        else if(source == btnViewCancel)
                        {
                                windowViews.dispose();
                                setEnabled(true);
                                show();
                        }
//--------------------------------END LISTENER VIEW --------------------------------
//-----------------------------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();
                        }
//-------------------------ENDED-LISTENER SELECT USER ------------------------
// listener start game
                        else if(source==btnMScoringQuit)
                        {
                                QuitOff();
                                setEnabled(true);
                                show();
                        }
// end listener start game
                }
//----------------END OF ACTIONLISTENER---------------------- END OF
ACTIONLISTENER ----------------END OF ACTIONLISTENER-----------
// fungsi untuk inisialisasi awal, terdiri dari load dan menampilkan user pada
current index ke 0
                public int initialize() {
                        try {
                                current = 0;
                                loadFile();
                                showPeople(current);
```

```
                    return 1;
                }catch(Exception e) {
        //JOptionPane.showMessageDialog(null,"Error in reading file");
                    JOptionPane.showMessageDialog(null, "Tidak
Ada User", "Warning", JOptionPane.ERROR_MESSAGE);
                    return 0;
                }
        }

// fungsi yang digunakan untuk meload file user.sav
        public void loadFile() throws IOException{
                peoplesTemp.removeAllElements();
                BufferedReader buff = new BufferedReader(new
FileReader("user.sav"));
                String line = null;
                while((line=buff.readLine())!=null) {
                        peoplesTemp.add(line);
                }
                buff.close();
        }

// fungsi digunakan untuk menampilkan isi dari user pada index tertentu -
digunakan pada halaman view user.
        public void showPeople(int index){
                String people = (String) peoplesTemp.elementAt(index);
                String split[] = people.split(";");
                VtxtNama.setText(split[0]);

//pengecekan male / female
```

```
                    if(split[1].equals("Male")) { VoptMale.setSelected(true); }
                    else { VoptFemale.setSelected(true); }

// show date user
                    VcomboMonth.setSelectedItem(split[2]);
                    VcomboDay.setSelectedItem( Integer.parseInt(split[3]) );
                    VcomboYear.setSelectedItem( Integer.parseInt(split[4]) );

//ph.setText(split[1]);

//email.setText(split[2]);
          }

//---VOID2-------------------------------------------------
//--VOID CEK LEVEL , START ON(untuk memulai permainan), QuitOff(untuk
selesai game)
          public void StartOn()
          {
//setEnabled(true);
// untuk inisialisasi variabel ulang
                    heroPosX =1;
                    heroPosY =1;
                    z=0;

                    CurrLv=1;

                    LockLv1 = 1;
                    LockLv2 = 2;
                    LockLv3 = 3;
```

```
            StepLv1 = 50;
            StepLv2 = 60;
            StepLv3 = 70;

            CurrScore = 0;
            CurrStep = 0;

            lblMScoringWarriorValue.setText(""+PlayedUser);
            lblMScoringStepValue.setText(""+StepLv1);
            lblMScoringScoreValue.setText(""+CurrScore);
            lblMScoringLevelValue.setText(""+CurrLv);

//LoadFileMap();
            LoadMap(CurrLv);
            show();
            windowStartGame.show();
            windowMenuScoring.show();
            windowPlayGameLv1.show();
}
public void QuitOff()
{
            windowStartGame.dispose();
            windowMenuScoring.dispose();
            windowPlayGameLv1.dispose();
            show();
            setEnabled(true);
}
```

```
        public void CekLevel()
        {

// 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,"Congratulat
ion, Level "+CurrLv+"
Accomplish","Message",JOptionPane.INFORMATION_MESSAGE);

        CurrLv=2;
        LoadMap(CurrLv);
        lblMScoringLevelValue.setText(""+CurrLv);
        lblMScoringStepValue.setText(""+StepLv2);

//System.out.println("CurrLv = "+CurrLv);
                                }
                        }
                        else if(CurrLv==2)
                        {
                        LockLv2--;
```

```
                    if(LockLv2==0)
                    {

            CurrStep=CurrStep+(60-StepLv2);
            CurrScore=CurrScore+(StepLv2*100);
            lblMScoringScoreValue.setText(""+CurrScore);

            JOptionPane.showMessageDialog(windowPlayGameLv1,"Congratulat
ion, Level "+CurrLv+"
Accomplish","Message",JOptionPane.INFORMATION_MESSAGE);

            CurrLv++;
            LoadMap(CurrLv);
            lblMScoringLevelValue.setText(""+CurrLv);
            lblMScoringStepValue.setText(""+StepLv3);

//System.out.println("CurrLv = "+CurrLv);
                                        }

                                }
                                else if(CurrLv==3)
                                {
                                LockLv3--;
                                if(LockLv3==0)
                                {
            CurrStep=CurrStep+(70-StepLv3);
            CurrScore=CurrScore+(StepLv3*100);
            lblMScoringScoreValue.setText(""+CurrScore);
//==WON WON
```

```
            JOptionPane.showMessageDialog(windowPlayGameLv1,"CONGRA
TULATION, "+PlayedUser+" YOU HAVE WON THE
GAME","Message",JOptionPane.INFORMATION_MESSAGE);
            setEnabled(true);

//-----------SAVE HIGH SCORE----------
//-----------SAVE DATA LAMA DULU
Vector VecHiScorex = new Vector();
                                                try
                                                {

BufferedReader buf = new BufferedReader(new FileReader("hiScore.sav"));
            String line = null;
            while((line=buf.readLine())!=null) {
            VecHiScorex.add(line);
            }
            buf.close();
            }catch(Exception ex) {
            JOptionPane.showMessageDialog(null,"Error 1");
            }

// ----Penyimpanan DATA USER ----------SAVE OTOMATIS ke "user.sav"-----
                                                try
                                                {
            PrintWriter p2 = new PrintWriter(new FileWriter("hiScore.sav"));


// menulis kembali data user sebelumnya
```

```
                    for(int i=0;i<VecHiScorex.size();i++)
                    {
                    String CurrentUser = (String) VecHiScorex.elementAt(i);
                    String split2[] = CurrentUser.split(";");

            p2.print(split2[0]+";");
            p2.print(split2[1]+";");
            p2.print(split2[2]);
            p2.println();
            }

// menambah data yang baru
            p2.print(PlayedUser+";");
            p2.print(CurrScore+";");
            p2.print(CurrStep);
            p2.println();
            p2.close();
            //loadFile();

//peoplesTemp.add(tempNamaUser+";"+tempGenderUser+";"+tempMonth+";"+
tempDay+";"+tempYear);

} catch(Exception ex) {JOptionPane.showMessageDialog(null,"Error 2"); }

//----------END SAVE DATA LAMA
//---------SORTING HIGH SCORE SEBELUM SAVE======
// -----------------BACA DATA User di "user.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) {
JOptionPane.showMessageDialog(null,"Error 3"); }

                                        String TempUsrx[] =  new String[6];
                                        int TempScrx[] = new int[6];
                                        String TempStepx[] =  new String[6];
                                        String TempSortUsrx = "";
                                        int TempSortScrx;
                                        String TempSortStepx =  "";
                                        for(int i=0;i<VecHiScore.size();i++)
                                        {
                        String CurrentUser2 = (String) VecHiScore.elementAt(i);
                        String split2x[] = CurrentUser2.split(";");
                        TempUsrx[i] = split2x[0];
                        TempScrx[i] = Integer.parseInt(split2x[1]);
                        TempStepx[i] = split2x[2];
                                        }


//--------------------SORTING ASCENDING
```

```
                    for(int i =0;i<VecHiScore.size();i++) {
                    for(int j=VecHiScore.size()-1;j>i;j--) {
                    if(TempScrx[j]>TempScrx[j-1]) {

            System.out.print(TempScrx[j]+"-"+TempScrx[j-1]);
            TempSortScrx = TempScrx[j];
            TempScrx[j] = TempScrx[j-1];
            TempScrx[j-1] = TempSortScrx;
            TempSortUsrx = TempUsrx[j];
            TempUsrx[j] = TempUsrx[j-1];
            TempUsrx[j-1] = TempSortUsrx;
            TempSortStepx = TempStepx[j];
            TempStepx[j] = TempStepx[j-1];
            TempStepx[j-1] = TempSortStepx;
            }
            System.out.println();
                    }
            }

//------------------END SORTING
//---Penyimpanan DATA USER LAGI setelah di sort----SAVE OTOMATIS ke
"user.sav"-----
                                        try
                                        {
            PrintWriter p2x = new PrintWriter(new FileWriter("hiScore.sav"));

// menulis kembali data user sebelumnya hanya TOP 5 yang ditulis
                                                for(int i=0;i<5;i++)
            {
```

```
            p2x.print(TempUsrx[i]+";");
            p2x.print(TempScrx[i]+";");
            p2x.print(TempStepx[i]);
            p2x.println();
            }
            p2x.close();
} catch(Exception ex) { JOptionPane.showMessageDialog(null,"Error 4"); }

//----------------END SORTING HIGH SCORE==========
            QuitOff();

//----------------END----SAVE HIGH SCORE-END----------
/=========WON WON
                                                                        }//
ENDED 'if LockLv==3'
}// ENDED 'cek level dan gembok yang sudah selesai'
            }

//--------------VOID LOAD FILE MAP-----------
            public void LoadFileMap()
            {
                        try
                        {
BufferedReader buf = new BufferedReader(new FileReader("map.txt"));
                                    int temp;
                                    int i=0;
                                    int j=0;
                                    z=0;
```

```
                while(true) {
                        temp=buf.read();
                        if(temp==-1) {break;}
                        else
                        {
if(Character.getNumericValue(temp)!=-1)
                                {
                                        if(z<100) {

mapData1[i][j] = Character.getNumericValue(temp);
                        j++;
                        if(j==10){ i++; j=0; }
                        }
                        else if(z<200)        {

mapData2[i][j] = Character.getNumericValue(temp);
                        j++;
                        if(j==10){ i++; j=0; }
                        }
                        else if(z<300)        {

mapData3[i][j] = Character.getNumericValue(temp);
                        j++;
                        if(j==10){ i++; j=0; }
                                        }
                                        z++;
                                }
                        }
                        if(i==10 && z==100) {i=0;}
```

```
                                        if(i==10 && z==200) {i=0;}
                                        if(i==10 && z==300) {i=0;}
                                }
                                buf.close();

                        } catch(Exception e) {
JOptionPane.showMessageDialog(null,"Error in reading file"); }

// inisialisasi map awal
                        if(CurrLv==1)
                        {
                                for(int i=0;i<10;i++) {
                                                for(int j=0;j<10;j++) {

        mapData[i][j]=mapData1[i][j];
                        if(mapData[i][j]==0)
                        {
                        lIsiMapLv1[i][j] = new JLabel(imgTanah);
                        }
                        else if(mapData[i][j]==1)
                        {
                        lIsiMapLv1[i][j] = new JLabel(imgBatu);
                        }
                        else if(mapData[i][j]==2)
                        {
                        lIsiMapLv1[i][j] = new JLabel(imgKunci);
                        }
                        else if(mapData[i][j]==3)
                        {
```

```
                    lIsiMapLv1[i][j] = new JLabel(imgGembokLock);
                    }
                    pPlayGamePapanLv1.add(lIsiMapLv1[i][j]);
                    }
          }
}// ENDED inisialisasi map
}

//----------------------
          public void StepOn()
          {

//------------------STEP
          if(StepLv1==1 || StepLv2==1 || StepLv3==1 )
          {

          JOptionPane.showMessageDialog(windowPlayGameLv1,"GAME
OVER, PLEASE TRY AGAIN "+PlayedUser,"GAME
OVER",JOptionPane.WARNING_MESSAGE);
          QuitOff();
                              }
// Pengurangan step
                              if(CurrLv==1)
                              {
                              StepLv1--;
                              lblMScoringStepValue.setText(""+StepLv1);
                              }
                              else if(CurrLv==2)
                              {
```

```
                                        StepLv2--;
                        lblMScoringStepValue.setText(""+StepLv2);
                                }
                                else if(CurrLv==3)
                                {
                                        StepLv3--;
                        lblMScoringStepValue.setText(""+StepLv3);
                                }
//-----------------END STEP
        }

//--------------------------------------VOID LOAD MAP
        public void LoadMap(int level)
        {

//System.out.println("CurrLv - 1 = "+(CurrLv-1)*100);
/*
// cek output
                                for(int i=0;i<10;i++)
                                {
                                                for(int j=0;j<10;j++)
                                                {
        System.out.print(mapData[i][j]);
                                                }
        System.out.print("\n");
                                        }
        System.out.println("total = "+z);

*/
```

```
//---------------------------PELETAKAN IMAGE-Frame Lv1--------------------------

        // LV1
                if(CurrLv==1)
                {
                        for(int i=0;i<10;i++) {
                                for(int j=0;j<10;j++) {

        mapData[i][j]=mapData1[i][j];
                if(mapData1[i][j]==0)           {
        lIsiMapLv1[i][j].setIcon(imgTanah);     }
                else if(mapData1[i][j]==1)      {
        lIsiMapLv1[i][j].setIcon(imgBatu);      }
                else if(mapData1[i][j]==2)      {
        lIsiMapLv1[i][j].setIcon(imgKunci);     }
                else if(mapData1[i][j]==3)      {
        lIsiMapLv1[i][j].setIcon(imgGembokLock);        }
                else if(mapData1[i][j]==4)      {
        mapData1[i][j]=3;
        lIsiMapLv1[i][j].setIcon(imgGembokLock);
                                                }

//lIsiMap[i][j].setBorder(BorderFactory.createLineBorder(Color.GRAY));
//pPlayGamePapanLv1.add(lIsiMapLv1[i][j]);
                }
                        }
                }// ENDED lv==1
                if(CurrLv==2)
                {
```

```
                        for(int i=0;i<10;i++) {
                                        for(int j=0;j<10;j++) {
            mapData[i][j]=mapData2[i][j];
            if(mapData2[i][j]==0)       {
            lIsiMapLv1[i][j].setIcon(imgTanah);     }
                        else if(mapData2[i][j]==1)     {
            lIsiMapLv1[i][j].setIcon(imgBatu);      }
                        else if(mapData2[i][j]==2)     {
            lIsiMapLv1[i][j].setIcon(imgKunci);     }
                        else if(mapData2[i][j]==3)     {
            lIsiMapLv1[i][j].setIcon(imgGembokLock);        }
                        else if(mapData2[i][j]==4)     {
            mapData2[i][j]=3;
            lIsiMapLv1[i][j].setIcon(imgGembokLock);        }

//lIsiMap[i][j].setBorder(BorderFactory.createLineBorder(Color.GRAY));

            //pPlayGamePapanLv1.add(lIsiMapLv1[i][j]);
                                        }
                        }
                }// ENDED lv==2
                // LV3
                if(CurrLv==3)
                {
                        for(int i=0;i<10;i++) {
                                        for(int j=0;j<10;j++) {

            mapData[i][j]=mapData3[i][j];
            if(mapData3[i][j]==0)       {
```

```
            lIsiMapLv1[i][j].setIcon(imgTanah);     }
                        else if(mapData3[i][j]==1)     {
            lIsiMapLv1[i][j].setIcon(imgBatu);       }
                        else if(mapData3[i][j]==2)     {
            lIsiMapLv1[i][j].setIcon(imgKunci);     }
                        else if(mapData3[i][j]==3)     {
            lIsiMapLv1[i][j].setIcon(imgGembokLock);        }
                        else if(mapData3[i][j]==4)     {
            mapData3[i][j]=3;
            lIsiMapLv1[i][j].setIcon(imgGembokLock);
            }

//lIsiMap[i][j].setBorder(BorderFactory.createLineBorder(Color.GRAY));

            //pPlayGamePapanLv1.add(lIsiMapLv1[i][j]);
                                                        }
                                    }
                        }// ENDED lv==3

//---------------------------------PELETAKAN IMAGE END----------------------------

                        heroPosX=1;
                        heroPosY=1;

            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroRIGHT);
//posisi awal
            }

//---END VOID2 -------
```

```
//-------KEY LISTENER
        public void keyTyped(KeyEvent e) {}
        public void keyReleased(KeyEvent e) {}
        public void keyPressed(KeyEvent e) {
                        int key = e.getKeyCode();
                        // ------------------------------ATAS
                        if(key==38)
                        {
                                StepOn();
                        //System.out.println("ATAS");
                        // untuk jalan kalo ketemu tanah baru bisa pindah
                        if(mapData[heroPosX-1][heroPosY]==0)
                                {
        lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                                        heroPosX--;
                                }
                        // kalo di atas hero ada kunci
                        else if(mapData[heroPosX-1][heroPosY]==2)
                        {
                        //jika posisi 2 diatasnya terdapat tanah
                        if(mapData[heroPosX-2][heroPosY]==0)
                        {
                        //ubah posisi mapdata setelah kunci digeser
                        mapData[heroPosX-1][heroPosY]=0;    // posisi
kunci sebelunya menjadi 0 (Tanah)
                        mapData[heroPosX-2][heroPosY]=2;    // posisi
kunci tergeser

        //ubah posisi hero & kunci
```

```
            lIsiMapLv1[heroPosX-2][heroPosY].setIcon(imgKunci);
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                                heroPosX--;
                        }

//jika posisi 2 diatasnya terdapat gembokterkunci
                                else if(mapData[heroPosX-2][heroPosY]==3)
                                                {
//hilangkan kunci karena telah masuk ke gembok
                                mapData[heroPosX-1][heroPosY]=0;    // posisi
kunci sebelunya menjadi 0 (Tanah)
//mapData[heroPosX-2][heroPosY]=2;  // posisi kunci tergeser
//ubah posisi hero menjadi posisi kunci
            lIsiMapLv1[heroPosX-2][heroPosY].setIcon(imgGembokOpen);
// ubah image icon posisi gembok menjadi gembok terbuka
            mapData[heroPosX-2][heroPosY]=4;    // mencegah agar gembok
yang sudah terbuka tidak bisa dimasukin kunci lagi
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);       // posisi
hero awal menjadi tanah
            heroPosX--;
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroUP);
                                CekLevel();
                                }
                        }
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroUP);
                                }

// ----------------------------ENDED ATAS
// ----------------------------BAWAH
```

```
                    else if(key==40)
                    {
                            StepOn();
        //System.out.println("BAWAH");
        if(mapData[heroPosX+1][heroPosY]==0)
                            {
        lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                heroPosX++;
                    }
                    else if(mapData[heroPosX+1][heroPosY]==2)
                    {
        if(mapData[heroPosX+2][heroPosY]==0)
                    {
        //ubah posisi mapdata setelah kunci digeser
        mapData[heroPosX+1][heroPosY]=0;   // posisi kunci sebelunya
menjadi 0 (Tanah)
        mapData[heroPosX+2][heroPosY]=2;   // posisi kunci tergeser
        //ubah posisi hero & kunci
        lIsiMapLv1[heroPosX+2][heroPosY].setIcon(imgKunci);
        lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                    heroPosX++;
                    }

        //jika posisi 2 dibawahnya terdapat gembokterkunci
                else if(mapData[heroPosX+2][heroPosY]==3)
                            {
        //hilangkan kunci karena telah masuk ke gembok
```

```
            mapData[heroPosX+1][heroPosY]=0;   // posisi kunci sebelunya
menjadi 0 (Tanah)
            //mapData[heroPosX-2][heroPosY]=2;  // posisi kunci tergeser
            //ubah posisi hero menjadi posisi kunci
            lIsiMapLv1[heroPosX+2][heroPosY].setIcon(imgGembokOpen);//
ubah image icon posisi gembok menjadi gembok terbuka
            mapData[heroPosX+2][heroPosY]=4;                  // mencegah agar
gembok yang sudah terbuka tidak bisa dimasukin kunci lagi
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);       // posisi
hero awal menjadi tanah
                                    heroPosX++;
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroDOWN);
                                    CekLevel();
                                                        }
                                            }
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroDOWN);


                                    }
// ------------------------------ENDED BAWAH

// ------------------------------KIRI
                                    else if(key==37)
                                    {
                                            StepOn();
                                    //System.out.println("KIRI");
                                    if(mapData[heroPosX][heroPosY-1]==0)
                                            {
```

```
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                                        heroPosY--;
                                }
                        else if(mapData[heroPosX][heroPosY-1]==2)
                                {

            if(mapData[heroPosX][heroPosY-2]==0)
                                {
            //ubah posisi mapdata setelah kunci digeser
                        mapData[heroPosX][heroPosY-1]=0;    // posisi kunci
sebelunya menjadi 0 (Tanah)
            mapData[heroPosX][heroPosY-2]=2;    // posisi kunci tergeser
            //ubah posisi hero & kunci
            lIsiMapLv1[heroPosX][heroPosY-2].setIcon(imgKunci);
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                                heroPosY--;
                                }
//jika posisi 2 dikirinya terdapat gembokterkunci
            else if(mapData[heroPosX][heroPosY-2]==3)
                        {
//hilangkan kunci karena telah masuk ke gembok
            mapData[heroPosX][heroPosY-1]=0;    // posisi kunci sebelunya
menjadi 0 (Tanah)
//mapData[heroPosX-2][heroPosY]=2;  // posisi kunci tergeser
//ubah posisi hero menjadi posisi kunci
            lIsiMapLv1[heroPosX][heroPosY-2].setIcon(imgGembokOpen);//
ubah image icon posisi gembok menjadi gembok terbuka
```

```
            mapData[heroPosX][heroPosY-2]=4;                // mencegah agar
gembok yang sudah terbuka tidak bisa dimasukin kunci lagi
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);       // posisi
hero awal menjadi tanah
                                    heroPosY--;
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroLEFT);
                                    CekLevel();
                                    }
                        }
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroLEFT);
            }
// ----------------------------ENDED KIRI

// ----------------------------KANAN
                                    else if(key==39)
                                    {
                                            StepOn();
            //System.out.println("KANAN");
            if(mapData[heroPosX][heroPosY+1]==0)
            {
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                        heroPosY++;
                        }
                        else if(mapData[heroPosX][heroPosY+1]==2)
                        {
            if(mapData[heroPosX][heroPosY+2]==0)
                        {
```

```
//ubah posisi mapdata setelah kunci digeser
            mapData[heroPosX][heroPosY+1]=0;   // posisi kunci sebelunya
menjadi 0 (Tanah)
            mapData[heroPosX][heroPosY+2]=2;   // posisi kunci tergeser
            //ubah posisi hero & kunci
            lIsiMapLv1[heroPosX][heroPosY+2].setIcon(imgKunci);
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);
                                    heroPosY++;
                        }
//jika posisi 2 dikanannya terdapat gembokterkunci
                        else if(mapData[heroPosX][heroPosY+2]==3)
                        {
//hilangkan kunci karena telah masuk ke gembok
            mapData[heroPosX][heroPosY+1]=0;   // posisi kunci sebelunya
menjadi 0 (Tanah)
//mapData[heroPosX-2][heroPosY]=2;  // posisi kunci tergeser
//ubah posisi hero menjadi posisi kunci
            lIsiMapLv1[heroPosX][heroPosY+2].setIcon(imgGembokOpen);//
ubah image icon posisi gembok menjadi gembok terbuka
            mapData[heroPosX][heroPosY-2]=4;                  // mencegah agar
gembok yang sudah terbuka tidak bisa dimasukin kunci lagi
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgTanah);       // posisi
hero awal menjadi tanah
                        heroPosY++;
            lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroRIGHT);
                        CekLevel();
```

```
        }//ENDED 'jika posisi 2 dikanannya terdapat gembokterkunci'

                                        }

        lIsiMapLv1[heroPosX][heroPosY].setIcon(imgHeroRIGHT);

                                }
                                // ------------------------------ENDED KANAN

        }

}
```