LAPORAN TUGAS BESAR
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK
SEMESTER GANJIL 2018-2019

BATTLESHIP

Disusun oleh:

Muhammad Ilham Fidatama (F1D016057)
Muhammad Malik Saputra (F1D016059)
Susilo Pandu Waskito (F1D016083)

Assisten pembimbing:

Chaerus Sulton

LABORATORIUM SISTEM CERDAS
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MATARAM

2018

LEMBAR PENGESAHAN
LAPORAN TUGAS BESAR
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Dikerjakan oleh:

Muhammad Ilham Fidatama (F1D016057)
Muhammad Malik Saputra (F1D016059)
Susilo Pandu Waskito (F1D016083)

Mengetahui
Dosen Pengampu,

(Royana Afwani, S.T., M.T.)
NIP: 19850707 201404 2 001

Koordinator Asisten,

(M. Habib Ali Akbar)
NIM: F1D 015 046

Kepala Laboratorium Praktikum,

(Ir. Sri Endang Anjarwani, M. Kom.)
NIP: 19660403 200604 2 001

Daftar Isi

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pada zaman sekarang ini, terdapat banyak sekali jenis-jenis *game*. Mulai dari *game* berbasis *desktop* dan game yang terkenal saat ini yaitu *game* berbasis *mobile*. *Game – game* tersebut ada yang *online* maupun *offline.* *Game* sendiri banyak yang diminati oleh berbagai kalangan mulai dari anak-anak hingga orang dewasa. Salah satu *game* contohnya adalah sebuah *game* sederhana yang disebut BattleShip.

BattleShip merupakan permainan yang menghindari rudal-rudal yang lewat dengan menggerakkan kapal sendiri dengan arah kiri kanan atas dan bawah. Pemain dapat mendapat skor dengan setiap melewati rudal yang dilalui akan bertambah 1 poin di setiap rudalnya. Jika permainan selesai maka pemain akan menyimpan nilai di *high score.*

PBO (Pemrograman Berorientasi Objek) adalah suatu konsep pemrograman yang memanfaatkan kelas dan objek untuk keperluan penulisan program. Salah satu contoh bahasa pemrograman dengan orientasi objek adalah *Java*. Lebih khususnya, ada aplikasi bernama *Netbeans* yang dapat digunakan untuk membuat suatu program *Java* dengan basis GUI (*Graphical User Interface*). Aplikasi ini dapat mempermudah *developer* untuk mengembangkan berbagai jenis program *Java* dengan basis GUI.

Karena itulah, di sini akan dibuat suatu program *Java* bernama *Battle Ship*, dengan memanfaatkan segala fitur yang ada dalam aplikasi *Netbeans.*

1.2 Deskripsi Aplikasi

Permainan BattleShip merupakan permainan yang terdiri dari satu pemain. Permainan BattleShip ini mengutamakan kecepatan dan ketelitian dalam bermain, inti utama permainan ini adalah pemain menghindari musuh untuk mencapai skor tertinggi.

Prosedur dari program yang akan dibuat dapat diuraikan sebagai berikut:

1. Program menampilkan tampilan utama dari program BattleShip. Menu utama terdiri dari pilihan: tombol mulai untuk memulai permainan baru, tombol papan

skor untuk menampilkan skor tertinggi dan tombol keluar untuk keluar dari permainan atau program akan berhenti.

2. Jika pemain menekan pilihan tombol mulai, maka akan tampil *window* baru yang menjadi tujuan utama dari program ini, yaitu permainan BattleShip. Permainan ini bertujuan untuk mencapai *high score* dengan menghindari rudal, agar tidak mengalami *game over*.
3. Jika pemain mengalami *game over*, maka akan tampil *window* baru untuk mengisi *username* dan apabila skor pemain tinggi makan akan ditampilkan skor pemain dalam menu *high score* dan tombol menu.
4. Jika pemain menekan tombol *high score*, maka akan tampil *window* baru yang di dalamnya berisi skor tertinggi dari pemain yang tersimpan dalam *database*. Namun jika pemain menekan tombol menu, akan menampilkan menu utama dari permainan BattleShip ini.
5. Jika pemain menekan pilihan *exit*, maka program akan terhenti.

1.3 Batasan Masalah

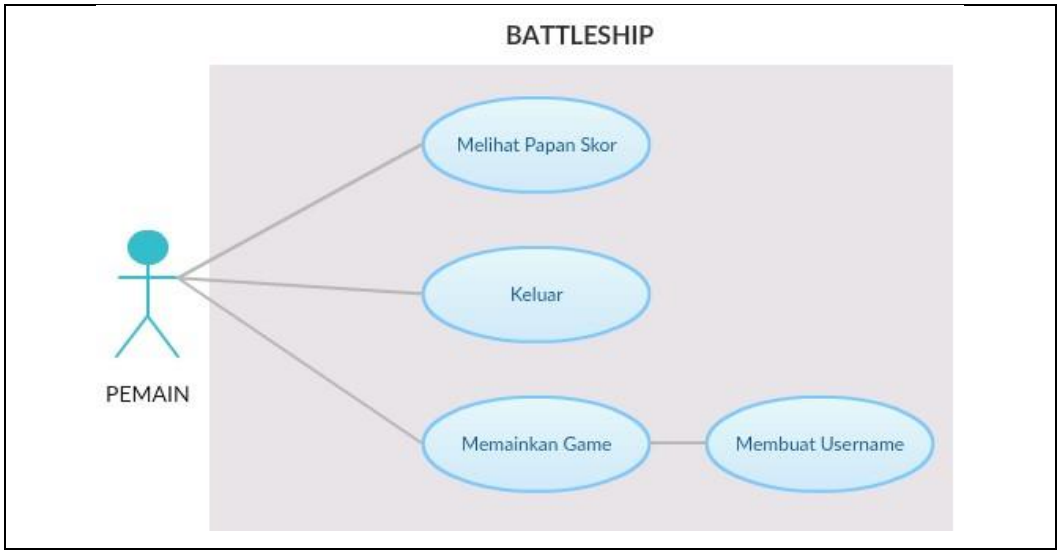
Batasan masalah dari aplikasi ini adalah pemain terdiri *single player* dan untuk memainkan *game* ini hanya perlu menggunakan *keyboard* dan menghindari musuh atau lawan dari pemain tersebut.

1.4 Tujuan

Tujuan dibuatnya program Java permainan BattleShip ini yaitu untuk mengimplementasikan pemograman berorientasi objek dan membuat variasi *game* bagi masyarakat luas.

BAB 2 ANALISA DAN DESAIN

2.1 Use Case Diagram



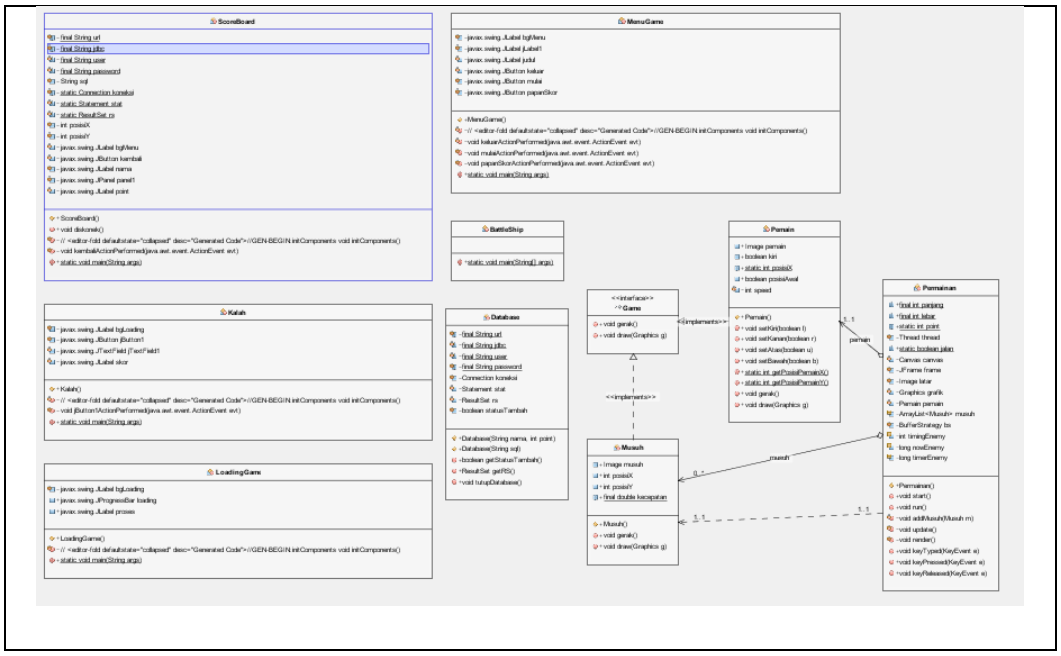
Gambar 1. *Use Case Diagram* BattleShip

Gambar 1 merupakan *Use Case Diagram* dari sisi pemain. Dimana *Use Case* ini berisi informasi mengenai hal-hal yang dapat dilakukan pemain.

Hal-hal yang dapat dilakukan *user*, yaitu:

1. Memainkan *game*
2. Membuat *username*
3. Melihat skor
4. Keluar dari permainan

2.2 Class Diagram



Gambar 2. *Class Diagram* BattleShip

2.3 Deskripsi Kelas

Gambar 2 merupakan *Class diagram* yang memiliki 10 *Class*. Deskripsi kelas di sini tidak mencantumkan *Method* dan variabel yang otomatis dibuat oleh NetBeans IDE, seperti *jLabel* dan sebagainya. Berikut ini merupakan daftar dan penjelasan dari kelas-kelas tersebut beserta *Method* dan variabel yang ada :

1. *Class Loading*

   Kelas ini digunakan untuk menyatukan data-data agar dapat digunakan.

2. *Class* MenuGame

   Kelas ini digunakan sebagai tampilan awal dalam permainan BattleShip.

   a. *Method*

   - *MenuGame()* : *Method* ini berfungsi sebagai *constructor* pada kelas *MenuGame*, menggunakan *acces modifier public*.
   - *JButton(Java.awt.event.ActionEvent evt)* : *Method* ini digunakan ketika pemain ingin memilih menu yang diinginkan dengan cara mengklik salah satu tombol yang ada pada menu.

3. *Class* BattleShip

   Kelas ini merupakan kelas yang paling penting, dikarenakan pada kelas ini permainan BattleShip dapat dijalankan dan dimainkan.

4. *Class* Pemain

   Kelas ini diguakan untuk dapat menggerakkan karakter kapal pada permainan BattleShip

   a. Atribut

      - *Kiri, kanan, atas, bawah :* Atribut ini digunakan untuk menentukan arah gerak kapal..
      - *PosisisX :* Atribut ini digunakan untuk menyimpan data secara horizontal.
      - *PosisisY :* Atribut ini digunakan untuk menyimpan data secara vertikal.
      - *PosisiAwal :* Atribut ini digunakan untuk menyimpan data posisi awal.
      - *Speed :* Atribut ini digunakan untuk mengatur kecepatan.

   b. *Method*

      - *Pemain() : Method* ini berfungsi sebagai *constructor* pada kelas *Pemain*, menggunakan *acces modifier public*.
      - *Void setKiri(boolean l) :* Merupakan *method* yang berfungsi untuk mengatur gerak kapal pemain ke kiri pada *form* permainan, menggunakan *acces modifier public*.
      - *Void setRight(boolean r) :* Merupakan *method* yang berfungsi untuk mengatur gerak kapal pemain ke kanan pada *form* permainan, menggunakan *acces modifier public*.
      - *Void setAtas(boolean u) :* Merupakan *method* yang berfungsi untuk mengatur gerak kapal pemain ke atas pada *form* permainan, menggunakan *acces modifier public*.
      - *Void setBawah(boolean b) :* Merupakan *method* yang berfungsi untuk mengatur gerak kapal pemain ke bawah pada *form* permainan, menggunakan *acces modifier public*.
      - *Static int getPosisiPemainX() :* Merupakan *method* yang berfungsi untuk mengatur posisi kapal pemain secara horizontal.
      - *Static int getPosisiPemainY() :* Merupakan *method* yang berfungsi untuk mengatur posisi kapal pemain secara vertikal.
      - *Void gerak() :* Merupakan *method* yang digunakan untuk menggerakkan kapal pemain.
      - *Void draw(Graphics g) :* Merupakan *method* yang berfungsi untuk mengatur *font*, ukuran, tulisan, panjang, lebar, warna setiap gambar yang

dimasukkan ke dalam program pada *form* permainan, menggunakan *acces modifier public*.

5. Kelas *Musuh*

   Kelas ini berfungsi untuk menghitung *score* pemain

   a. Atribut

   - *posisiX* : Atribut ini digunakan untuk mengatur posisi kapal secara horizontal.
   - *posisiY* : Atribut ini digunakan untuk mengatur posisi kapal secara vertikal.
   - *kecepatan* : Atribut ini digunakan untuk.mengatur kecepatan kapal pemain.

   b. *Method*

   - *musuh()* : *Method* ini berfungsi sebagai *constructor* pada kelas *Musuh*, menggunakan *acces modifier public*.
   - *void gerak() : Method* ini digunakan untuk mengatur gerakkan rudal.
   - *void draw(Graphics g) :* Merupakan *method* yang berfungsi untuk mengatur *font*, ukuran, tulisan, panjang, lebar, warna setiap gambar yang dimasukkan ke dalam program pada *form* permainan, menggunakan *acces modifier public*.

6. Kelas *Permainan*

   Kelas ini digunakan sebagai tempat menjalankan permainan, yang mana pada kelas ini karakter pada permainan BattleShip dapat digerakkan.

   a. Atribut

   - *panjang* : Atribut ini digunakan untuk mengatur panjang papan permainan.
   - *lebar* : Atribut ini digunakan untuk mengatur lebar papan permainan.
   - *point* : Atribut ini digunakan untuk menyimpan skor awal pada permainan.
   - *jalan* : Atribut ini digunakan untuk menjalankan kapal pemain.

   b. *Method*

   - *permainan()* : Merupakan *method* yang digunakan untuk mengatur posisi kapal dan rudal saat program dijalankan.
   - *void start() :* Merupakan *method* yang berfungsi untuk proses *Thread* pada program untuk memulai permainan.

- *void run()* : Merupakan *method* yang berfungsi untuk proses *Thread* pada program untuk menjalankan permainan.
- *void addMusuh(Musuh M) :* Merupakan *method* yang berfungsi untuk menambahkan karakter rudal pada permainan.
- *void update()* : Merupakan *method* yang berfungsi untuk mengatur pergerakan rudal.
- *void render() :* Merupakan *method* yang menggambarkan *gameplay* yang dibutuhkan.
- *void keyTyped(KeyEvent e)* : Merupakan *method* yang berfungsi untuk mengatur ketikan *keyboard*.
- *void keyPressed(KeyEvent e) :* Merupakan *method* yang berfungsi untuk mengatur *keyboard* mana yang seharusnya ditekan.
- *void keyReleased(KeyEvent e)* : Merupakan *method* yang berfungsi untuk mengatur *keyboard* mana yang dilepaskan.

7. Kelas *Database*

a. Atribut

- *koneksi* : Atribut yang digunakan sebagai variable untuk menyimpan URL yang akan di koneksikan ke *database*, menggunakan *acces modifier default*..
- *statusTambah* : Atribut ini digunakan untuk menyimpan nilai pemain yang baru.
- *rs* : Atribut ini digunakan untuk mengambil hasil eksekusi *query*.

b. *Method*

- *Database(String nama, int point)* : Merupakan *method* yang berfungsi untuk memperbaharui data pada *database.*
- *Database(String sql)* : Merupakan *method* yang berfungsi untuk mengambil data.
- *Boolean getStatusTambah(String nama, int point)* : Merupakan *method* yang berfungsi untuk
- *ResultSet getRS()* : Merupakan *method* yang berfungsi untuk mengambil hasil eksekusi *query.*
- *Void tutupDatabase()* : Merupakan *method* yang berfungsi untuk mengakhiri koneksi ke *database.*

8. Kelas *Kalah*

   Kelas ini berfungsi untuk menyimpan *username* pemain saat mereka kalah.

   a. *Method*

      - *kalah()* : *Method* ini berfungsi sebagai *constructor* pada kelas *Kalah*, menggunakan *acces modifier public*.
      - *JButton(Java.awt.event.ActionEvent evt) :* Merupakan *method* yang digunakan untuk membuat tombol untuk menyimpan data pemain.

9. Kelas *ScoreBoard*

   Kelas ini berfungsi untuk menampilkan *score* pemain tertinggi.

   a. Atribut

      - *posisiX* : Atribut ini digunakan untuk menyimpan posisi secara horizontal.
      - *posisiY* : Atribut ini digunakan untuk menyimpan posisi secara vertikal.

   b. *Method*

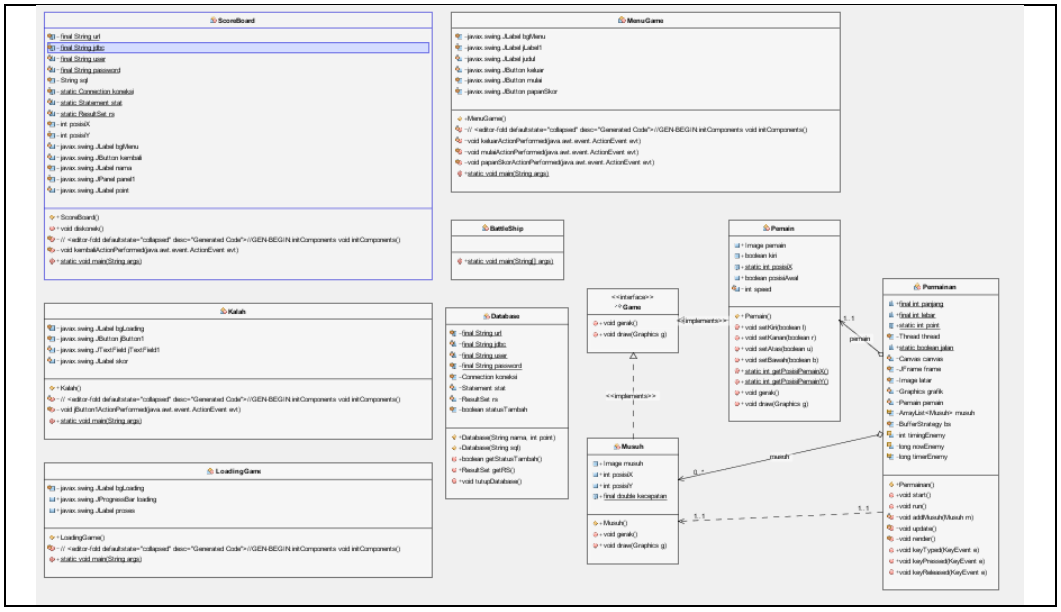
      - *ScoreBoard()* : Merupakan *method* yang berfungsi untuk menampilkan nama pemain dan skor yang tertinggi.
      - *Void disconek()* : Merupakan *method* yang berfungsi untuk menghentikan koneksi ke *database.*
      - *Void kembali()* : Merupakan *method* yang berfungsi untuk menghubungkan kembali ke dalam MenuGame.

10. Kelas *BattleShip*

    Kelas ini merupakan kelas utama yang akan mengatur alur jalannya program pada permainan BattleShip.

BAB 3 IMPLEMENTASI

3.1 Implementation Class Diagram



Gambar 3. *Class Diagram* Battleship

Dari diagram kelas pada gambar di atas, dibuat ke dalam bentuk program Java, dengan jumlah kelas 10 dan dengan keterhubungan tertentu antar kelas.

3.2 Implementasi Konsep Pemrograman Berorientasi Objek

Berikut ini penjelasan mengenai penggunaan konsep PBO pada program Battleship sebagai berikut:

1. Abstraksi

   Abstraksi adalah suatu metode untuk melakukan trans*form*asi dari suatu kasus ke dalam bentuk entitas dan relasinya[1]. Pada program Battleship ini, awalnya dibuat entitas-entitas awal yang kira-kira dibutuhkan dalam program, seperti: tampilan awal, tampilan saat bermain, tampilan saat melihat skor tertinggi. Dari sana kemudian dikembangkan menjadi beberapa kelas baru sesuai dengan kebutuhan.

2. Enkapsulasi

   Pada program Battleship, dilakukan proses enkapsulasi. Contohnya, dibuat objek dari kelas *MenuGame.* Kemudian dari objek tersebut dipanggil

fungsi *setVisible*(*true*), dimana tidak perlu diketahui bagaimana kode programnya.[1]

3. Pewarisan (*Inheritance*)

   Dalam program ini, semua kelas yang menggunakan tampilan GUI menggunakan kelas bawaaan dari Java yaitu *JFrame, JPanel, Thread*. Pewarisan ini dapat dilihat pada kelas *Diagram*, dimana hubungan antar kelas dinyatakan dengan panah putih. Panah mengarah dari kelas yang mewarisi ke kelas induk.

4. *Polymorphism*

   Polymorphism merupakan kondisi dimana beberapa *Method* dengan nama yang sama dapat memiliki aksi yang berbeda. *Polymorphism* ini dibagi dua, yaitu *overriding* dan *overloading.*

   Overriding merupakan *Method* dengan nama yang sama, tipe data yang sama, dan parameter yang sama dengan yang ada pada kelas induk, namun memiliki isi (*statement*) yang berbeda[2]. Penggunaan *Method* tersebut terdapat pada kelas *Showboard*. Sedangkan *Overloading* adalah membuat *Method* dengan nama yang sama, namun dengan parameter yang berbeda.

5. *Abstract Class*

   Kelas abstrak adalah kelas yang minimal memiliki satu *Method* abstrak. Kelas abstrak yang ada dalam program Battleship ini adalah yaitu pada kelas *Game*.

6. *Interface*

   Pada program BattleShip, menggunakan *interface* bawaan dari Java seperti, *ActionListener* dan *KeyListener*

7. GUI

   Pada program BattleShip, GUI dibuat menggunakan *tool pallete* yang ada di *NetBeans* IDE. GUI membuat program menjadi lebih atraktif dan interaktif. Kelas yang menerapkan GUI yaitu *LoadingGame, MenuGame, ScoreBoard* dan *Kalah.*

BAB 4 PENUTUP

4.1 Kesimpulan

Dari pembuatan tugas akhir praktikum Pemrograman Berorientasi Objek, dapat diambil kesimpulan bahwa:

1. Kuliah Pemrograman Berorientasi Objek ini, menuntut mahasiswa agar dapat memahami konsep dan dapat melakukan pengimplementasian dalam Pemrograman Berorientasi Objek.
2. Pada pembuatan *project* akhir, dilakukan pengimplementasian semua konsep dari seluruh modul yang telah dikerjakan saat praktikum.

4.2 Saran

Dalam pembuatan permainan Battleship ini diharapkan agar ditambahkan variasi kapal, level permainan dan di harapkan agar semua anggota kelompok juga harus lebih aktif dalam pembuatan *project* akhir ini, sehingga hasilnya dapat lebih maksimal.

DAFTAR PUSTAKA

[1] Ridho Barakbah, Ali. 2006. Pemrograman Berbasis Obyek. Institut Teknologi Sepuluh Nopember.

[2] Horstmann, Cay S. and Cornell, Gary. 2007. *Core Java, Volume 1: Fundamentals*. Prentice Hall PTR.

[3] Nugroho, Adi. 2004. *Pemrograman Berorientasi Objek.* In*form*atika: Bandung.

LAMPIRAN

BattleShip.java

```
package battleship;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author ASUS R.O.G
 */
public class BattleShip {

    public static void main(String[] args) {
        LoadingGame lg = new LoadingGame();
        lg.setVisible(true);

        try {
            for (int i = 0; i <= 100; i++) {
                Thread.sleep(10);
                lg.proses.setText("LOAD&ING DATA
"+Integer.toString(i)+"%");
                lg.loading.setValue(i);
            }
        } catch (InterruptedException ex) {

Logger.getLogger(MenuGame.class.getName()).log(Level.SEVERE, null,
ex);
        }
        MenuGame bs = new MenuGame();
        bs.setVisible(true);
        lg.dispose();
    }

}
```

LoadingGame.java

```
package battleship;

/**
 *
 * @author ASUS R.O.G
 */
public class LoadingGame extends javax.swing.JFrame {

    public LoadingGame() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        proses = new javax.swing.JLabel();
        loading = new javax.swing.JProgressBar();
        bgLoading = new javax.swing.JLabel();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
;
        setUndecorated(true);
        getContentPane().setLayout(null);

        proses.setBackground(new java.awt.Color(255, 255, 255));
        proses.setFont(new java.awt.Font("Times New Roman", 1,
18)); // NOI18N
        proses.setForeground(new java.awt.Color(255, 255, 255));

proses.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        proses.setLabelFor(loading);
        getContentPane().add(proses);
        proses.setBounds(450, 580, 200, 30);

        loading.setBackground(new java.awt.Color(255, 255, 255));
        loading.setFont(new java.awt.Font("Times New Roman", 1,
18)); // NOI18N
        loading.setForeground(new java.awt.Color(0, 255, 0));
        loading.setToolTipText("LOADING");
        loading.setBorderPainted(false);
        loading.setString("");
        loading.setStringPainted(true);
        getContentPane().add(loading);
        loading.setBounds(50, 550, 1000, 30);

        bgLoading.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/loading1.jpg")
)); // NOI18N
        getContentPane().add(bgLoading);
        bgLoading.setBounds(0, 0, 1100, 650);

        setSize(new java.awt.Dimension(1100, 650));
        setLocationRelativeTo(null);
    }// </editor-fold>

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and
feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf
.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LoadingGame.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(LoadingGame.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LoadingGame.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(LoadingGame.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LoadingGame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel bgLoading;
    public javax.swing.JProgressBar loading;
    public javax.swing.JLabel proses;
    // End of variables declaration
}
```

MenuGame.java

```
package battleship;

import java.io.IOException;
import java.sql.SQLException;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author ASUS R.O.G
 */
public class MenuGame extends javax.swing.JFrame {

    public MenuGame() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        judul = new javax.swing.JLabel();
        keluar = new javax.swing.JButton();
        papanSkor = new javax.swing.JButton();
        mulai = new javax.swing.JButton();
        bgMenu = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
;
```

```
        setUndecorated(true);
        getContentPane().setLayout(null);

        jLabel1.setText("jLabel1");
        getContentPane().add(jLabel1);
        jLabel1.setBounds(100, 360, 34, 14);

        judul.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/TOMBOL/judul.p
ng"))); // NOI18N
        getContentPane().add(judul);
        judul.setBounds(345, 30, 410, 53);

        keluar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/TOMBOL/keluar.
png"))); // NOI18N
        keluar.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                keluarActionPerformed(evt);
            }
        });
        getContentPane().add(keluar);
        keluar.setBounds(462, 280, 176, 60);

        papanSkor.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/TOMBOL/papanSk
or.png"))); // NOI18N
        papanSkor.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                papanSkorActionPerformed(evt);
            }
        });
        getContentPane().add(papanSkor);
        papanSkor.setBounds(580, 180, 250, 60);

        mulai.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/TOMBOL/mulai.p
ng"))); // NOI18N
        mulai.setMargin(new java.awt.Insets(0, 0, 0, 0));
        mulai.setMaximumSize(new java.awt.Dimension(300, 110));
        mulai.setMinimumSize(new java.awt.Dimension(250, 60));
        mulai.setPreferredSize(new java.awt.Dimension(250, 60));
        mulai.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                mulaiActionPerformed(evt);
            }
        });
        getContentPane().add(mulai);
        mulai.setBounds(270, 180, 250, 60);

        bgMenu.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/Background.jpg
"))); // NOI18N
        getContentPane().add(bgMenu);
        bgMenu.setBounds(0, 0, 1100, 650);
```

```
        setSize(new java.awt.Dimension(1100, 650));
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void keluarActionPerformed(java.awt.event.ActionEvent
evt) {
        System.exit(0);
    }

    private void mulaiActionPerformed(java.awt.event.ActionEvent
evt) {
        try {
            Permainan play = new Permainan();
            play.start();
        } catch (IOException ex) {

Logger.getLogger(MenuGame.class.getName()).log(Level.SEVERE, null,
ex);
        }
        dispose();

    }

    private void
papanSkorActionPerformed(java.awt.event.ActionEvent evt) {
        ScoreBoard sc = new ScoreBoard();
        sc.setVisible(true);
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and
feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf
.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MenuGame.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MenuGame.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
```

```
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MenuGame.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MenuGame.class.getName()).log(ja
va.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MenuGame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel bgMenu;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel judul;
    private javax.swing.JButton keluar;
    private javax.swing.JButton mulai;
    private javax.swing.JButton papanSkor;
    // End of variables declaration
}
```

Musuh.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package battleship;

import java.awt.Graphics;
import java.awt.Image;
import java.awt.Rectangle;
import java.util.Random;
import javax.swing.ImageIcon;

/**
 *
 * @author ASUS R.O.G
 */
public class Musuh extends Game{
    public Image musuh;
    public int posisiX=1100;
    public int posisiY=62;
    public static final double kecepatan=5;

    public Musuh(){
        posisiY = (int) (Math.random() * 600);
        musuh = new
ImageIcon(this.getClass().getResource("/warfare/roket.png")).getIma
ge();
    }

    @Override
    public void gerak() {
```

```
        posisiX -= kecepatan;
        Rectangle musuh = new Rectangle(posisiX, posisiY, 41, 33);
        Rectangle pemain = new Rectangle(Pemain.getPosisiPemainX(),
Pemain.getPosisiPemainY(),210,35);

        if (musuh.intersects(pemain)) {
            Permainan.jalan=false;
        }
    }

    @Override
    public void draw(Graphics g) {
        g.drawImage(musuh, posisiX, posisiY, 41, 33, null);
    }
}
```

Pemain.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package battleship;

import java.awt.Graphics;
import java.awt.Image;
import javax.swing.ImageIcon;

/**
 *
 * @author ASUS R.O.G
 */
public class Pemain extends Game{
    public Image pemain;
    public boolean kiri, kanan, atas, bawah; //untuk menentukan
arah
    public static int posisiX=0, posisiY=250;
    public boolean posisiAwal=true, setKiri, setKanan;
    private final int speed=3;

    public Pemain(){
        pemain = new
ImageIcon(this.getClass().getResource("/warfare/battleship.png")).g
etImage();
    }

    public void setKiri(boolean l){
        kiri = l;
    }
    public void setKanan(boolean r){
        kanan = r;
    }
    public void setAtas(boolean u){
        atas = u;
    }
    public void setBawah(boolean b){
        bawah = b;
    }

    public static int getPosisiPemainX(){
        return posisiX;
```

```
    }

    public static int getPosisiPemainY(){
        return posisiY;
    }

    @Override
    public void gerak() {
        if (kiri){
            posisiX-=speed;
            setKiri=true;
            setKanan=false;
            posisiAwal=false;
        }
        if (kanan){
            posisiX+=speed;
            setKiri=false;
            setKanan=true;
            posisiAwal=false;
        }
        if (atas){
            posisiY-=speed;
        }
        if (bawah){
            posisiY+=speed;
        }
        //kondisi-kondisi di bawah ini digunakan agar objek nya
tidak keluar dari frame
        if (posisiX<=0){
            kiri=false;
            posisiX=0;
        }

        if (posisiX>=1100){
            kanan=false;
            posisiX=1100;
        }

        if (posisiY<=0){
            atas=false;
            posisiY=0;
        }

        if (posisiY>=650){
            bawah=false;
            posisiY=650;
        }
    }

    @Override
    public void draw(Graphics g) {
        g.drawImage(pemain, posisiX, posisiY, null);
    }

}
```

Permainan.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package battleship;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.image.BufferStrategy;
import java.io.IOException;
import java.util.ArrayList;
import javax.swing.ImageIcon;
import javax.swing.JFrame;

/**
 *
 * @author ASUS R.O.G
 */
public class Permainan implements Runnable, KeyListener{
    private static final int panjang=1100;
    private static final int lebar=650;
    public static int point=0;
    private Thread thread;
    public static boolean jalan;
    private final Canvas canvas;
    private final JFrame frame;
    private final Image latar;
    private Graphics grafik;
    private final Pemain pemain;
    ArrayList<Musuh> musuh = new ArrayList<Musuh>();
    private BufferStrategy bs; //agar game tidak patah-patah

    private int timingEnemy=1000; //kecepatan dibentuknya objek
    private long nowEnemy = System.nanoTime(); //waktu untuk
perpindahan posisi objek
    private long timerEnemy = 1000000; //pergerakan objek

    public Permainan()throws IOException{
        jalan=true;
        frame = new JFrame("BattleShip");
        frame.setSize(panjang, lebar);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setUndecorated(true);
        frame.setResizable(false);
        frame.setVisible(true);
        canvas = new Canvas();
        canvas.setPreferredSize(
                new Dimension(panjang, lebar)
        );
        canvas.setMinimumSize(
                new Dimension(panjang, lebar)
        );
        canvas.setFocusable(true);
        frame.add(canvas);
        canvas.addKeyListener(this);
```

```
        latar = new
ImageIcon(this.getClass().getResource("/image/latar.gif")).getImage
(); // NOI18N
        pemain = new Pemain();
    }

    public synchronized void start(){
        if (point !=0){ //untuk memulai skor dari 0 kembali
            point = 0;
        }
        thread = new Thread(this); //membuat thread
        thread.start(); //menjalankan thread
    }

    @Override
    public void run() {
        double timePerTick = 1000000000/60;
        double delta = 0;
        long now;
        long lastTime = System.nanoTime();

        while(jalan){ //selama boolean jalan true thread akan tetap
dijalankan
            now = System.nanoTime();
            delta += (now - lastTime)/timePerTick;
            lastTime = now;
            if(delta >= 1){
                update(); //update posisi objek ikan
                render(); //menggambar objek ikan
                delta --;
            }
        }

        Kalah lose = new Kalah();
        Pemain.posisiX=0;
        Pemain.posisiY=250;
        lose.setVisible(true);
        frame.dispose();

    }

    private void addMusuh(Musuh m){
        musuh.add(m);
    }

    private void update(){ //method untuk pergerakan ikan
        long waktu = (System.nanoTime()-nowEnemy)/timerEnemy;

        for(int i=0; i<musuh.size(); i++){ //untuk gerakin ikan
musuh
             musuh.get(i).gerak();
        }

        if (waktu>timingEnemy){ //kecepatan untuk buat objek
            addMusuh(new Musuh());
            nowEnemy = System.nanoTime();
//            timingEnemy = timingEnemy-2;
        }

        for(int i = 0; i < musuh.size();i++){ // remove arraylist
dari ikan musuh
```

```
            if(musuh.get(i).posisiX <= -50){
                musuh.remove(i);
                point+=8;
            }
        }
    }

    private void render(){ //menggambar gameplay yg dibutuhkan

        bs = canvas.getBufferStrategy(); //canvas diberikan
buffered strategy agar tidak patah-patah

        if(bs == null){ //kondisi ketika buffered strategy masih
kosong
            canvas.createBufferStrategy(2);
            return;
        }

        grafik = bs.getDrawGraphics();
        grafik.drawImage(latar, 0, 0, null);

        pemain.draw(grafik); //menggambar ikan pemain dalam
frame/canvas

        for(int i=0; i<musuh.size(); i++){ //menggambar ikan musuh
            musuh.get(i).draw(grafik);
        }

        grafik.setColor(Color.white);
        grafik.setFont(
            new Font("Bebas Neue", Font.PLAIN, 50)
        );
        grafik.drawString(""+point, 550, 50); //menampilkan skor
diatas layar

        pemain.gerak(); //mengerakan posisi pemain
        bs.show();  //menjalankan buffered strategy

    }

    @Override
    public void keyTyped(KeyEvent e) {}

    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode()==KeyEvent.VK_A) {
            pemain.setKiri(true);
        }
        else if(e.getKeyCode()==KeyEvent.VK_D){
            pemain.setKanan(true);
        }
        else if(e.getKeyCode()==KeyEvent.VK_W){
            pemain.setAtas(true);
        }
        else if(e.getKeyCode()==KeyEvent.VK_S){
            pemain.setBawah(true);
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {
```

```
        if (e.getKeyCode()==KeyEvent.VK_A) {
            pemain.setKiri(false);
        }
        else if(e.getKeyCode()==KeyEvent.VK_D){
            pemain.setKanan(false);
        }
        else if(e.getKeyCode()==KeyEvent.VK_W){
            pemain.setAtas(false);
        }
        else if(e.getKeyCode()==KeyEvent.VK_S){
            pemain.setBawah(false);
        }
    }
}
```

Kalah.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package battleship;

import javax.swing.SwingConstants;

/**
 *
 * @author ASUS R.O.G
 */
public class Kalah extends javax.swing.JFrame {

    /**
     * Creates new form LoadingGame
     */
    public Kalah() {
        initComponents();
        skor.setHorizontalAlignment(SwingConstants.CENTER);
        skor.setText(Integer.toString(Permainan.point));
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        skor = new javax.swing.JLabel();
        namaPemain = new javax.swing.JTextField();
        simpan = new javax.swing.JButton();
        bgLoading = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
;
        setUndecorated(true);
        getContentPane().setLayout(null);
```

```
        skor.setFont(new java.awt.Font("Bebas Neue", 1, 100)); //
NOI18N
        skor.setForeground(new java.awt.Color(255, 255, 255));
        skor.setFocusable(false);

skor.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        getContentPane().add(skor);
        skor.setBounds(410, 200, 300, 150);

        namaPemain.setFont(new java.awt.Font("Bebas Neue", 3, 70));
// NOI18N
        getContentPane().add(namaPemain);
        namaPemain.setBounds(385, 370, 350, 100);

        simpan.setBackground(new java.awt.Color(0, 255, 0));
        simpan.setFont(new java.awt.Font("Bebas Neue", 3, 30)); //
NOI18N
        simpan.setText("SIMPAN");
        simpan.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                simpanActionPerformed(evt);
            }
        });
        getContentPane().add(simpan);
        simpan.setBounds(490, 500, 150, 50);

        bgLoading.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/kalah.jpg")));
// NOI18N
        getContentPane().add(bgLoading);
        bgLoading.setBounds(0, 0, 1100, 650);

        setSize(new java.awt.Dimension(1100, 650));
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void simpanActionPerformed(java.awt.event.ActionEvent
evt) {
        String nama = namaPemain.getText();
        int score = Integer.parseInt(skor.getText());
        Database db = new Database(nama, score);
        MenuGame mg = new MenuGame();
        mg.setVisible(true);
        this.dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and
feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf
.html
```

```
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Kalah.class.getName()).log(java.
util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Kalah.class.getName()).log(java.
util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Kalah.class.getName()).log(java.
util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Kalah.class.getName()).log(java.
util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Kalah().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel bgLoading;
    private javax.swing.JTextField namaPemain;
    private javax.swing.JButton simpan;
    private javax.swing.JLabel skor;
    // End of variables declaration
}
```

ScoreBoard.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package battleship;

import java.awt.Font;
import java.io.IOException;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JLabel;

/**
 *
 * @author ASUS R.O.G
 */
public class ScoreBoard extends javax.swing.JFrame {
    private static final String url =
"jdbc:mysql://localhost/Battleship";
    private static final String jdbc = "com.mysql.jdbc.Driver";
    private static final String user = "root";
    private static final String password = "";
    private final String sql = "SELECT * FROM pemain ORDER BY point
DESC LIMIT 3";

    private static Connection koneksi;
    private static Statement stat;
    private static ResultSet rs;

    private int posisiX = 20;
    private int posisiY = 30;
    public ScoreBoard() {
        int i=0;
        initComponents();

        try {
            Database db = new Database(sql);
            rs = db.getRS();
            while(rs.next()){
                if (i==0) {
                    nama.setText(rs.getString("nama"));
                    point.setText(rs.getString("point"));
                }else if (i==1) {
                    nama1.setText(rs.getString("nama"));
                    point1.setText(rs.getString("point"));
                }else if (i==2) {
                    nama2.setText(rs.getString("nama"));
                    point2.setText(rs.getString("point"));
                }
                i++;
            }
        } catch (SQLException ex) {

Logger.getLogger(ScoreBoard.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    public void diskonek(){
        try {
            stat.close();
            koneksi.close();
        } catch (SQLException ex) {

Logger.getLogger(ScoreBoard.class.getName()).log(Level.SEVERE,
null, ex);
```

```
        }
    }

    /**
     * This method is called from within the constructor to
initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        kembali = new javax.swing.JButton();
        panel1 = new javax.swing.JPanel();
        nama = new javax.swing.JLabel();
        nama1 = new javax.swing.JLabel();
        nama2 = new javax.swing.JLabel();
        point = new javax.swing.JLabel();
        point1 = new javax.swing.JLabel();
        point2 = new javax.swing.JLabel();
        bgMenu = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
;
        setUndecorated(true);
        getContentPane().setLayout(null);

        kembali.setFont(new java.awt.Font("Rosewood Std Regular",
3, 36)); // NOI18N
        kembali.setForeground(new java.awt.Color(0, 0, 255));
        kembali.setText("KEMBALI");
        kembali.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                kembaliActionPerformed(evt);
            }
        });
        getContentPane().add(kembali);
        kembali.setBounds(10, 10, 190, 60);

        panel1.setBackground(new java.awt.Color(255, 255, 255));
        panel1.setFocusable(false);
        panel1.setOpaque(false);
        panel1.setLayout(null);

        nama.setBackground(new java.awt.Color(255, 255, 255));
        nama.setFont(new java.awt.Font("Bebas Neue", 0, 50)); //
NOI18N

nama.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        nama.setLabelFor(panel1);
        panel1.add(nama);
        nama.setBounds(20, 20, 230, 70);

        nama1.setBackground(new java.awt.Color(255, 255, 255));
        nama1.setFont(new java.awt.Font("Bebas Neue", 0, 50)); //
NOI18N
```

```
nama1.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        nama1.setLabelFor(panel1);
        panel1.add(nama1);
        nama1.setBounds(20, 90, 230, 70);

        nama2.setBackground(new java.awt.Color(255, 255, 255));
        nama2.setFont(new java.awt.Font("Bebas Neue", 0, 50)); //
NOI18N

nama2.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        nama2.setLabelFor(panel1);
        panel1.add(nama2);
        nama2.setBounds(20, 180, 230, 70);

        point.setBackground(new java.awt.Color(255, 255, 255));
        point.setFont(new java.awt.Font("Bebas Neue", 0, 50)); //
NOI18N

point.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        panel1.add(point);
        point.setBounds(270, 20, 230, 70);

        point1.setBackground(new java.awt.Color(255, 255, 255));
        point1.setFont(new java.awt.Font("Bebas Neue", 0, 50)); //
NOI18N

point1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        panel1.add(point1);
        point1.setBounds(270, 90, 230, 70);

        point2.setBackground(new java.awt.Color(255, 255, 255));
        point2.setFont(new java.awt.Font("Bebas Neue", 0, 50)); //
NOI18N

point2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        panel1.add(point2);
        point2.setBounds(270, 180, 230, 70);

        getContentPane().add(panel1);
        panel1.setBounds(300, 180, 500, 290);

        bgMenu.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/image/scoreboard.jpg
")));// NOI18N
        getContentPane().add(bgMenu);
        bgMenu.setBounds(0, 0, 1100, 650);

        setSize(new java.awt.Dimension(1100, 650));
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void kembaliActionPerformed(java.awt.event.ActionEvent
evt) {
        MenuGame mg = new MenuGame();
        mg.setVisible(true);
        this.dispose();
    }

    /**
     * @param args the command line arguments
```

```
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and
feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf
.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ScoreBoard.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ScoreBoard.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ScoreBoard.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ScoreBoard.class.getName()).log(
java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>


        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ScoreBoard().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel bgMenu;
    private javax.swing.JButton kembali;
    private javax.swing.JLabel nama;
    private javax.swing.JLabel nama1;
    private javax.swing.JLabel nama2;
    private javax.swing.JPanel panel1;
    private javax.swing.JLabel point;
    private javax.swing.JLabel point1;
    private javax.swing.JLabel point2;
    // End of variables declaration
}
```

Database.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package battleship;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author ASUS R.O.G
 */
public class Database {
    private static final String url =
"jdbc:mysql://localhost/Battleship";
    private static final String jdbc = "com.mysql.jdbc.Driver";
    private static final String user = "root";
    private static final String password = "";

    private Connection koneksi;
    private Statement stat;
    private ResultSet rs;
    private boolean statusTambah;

    public Database(String nama, int point){ //untuk menambah data
        String sql="INSERT INTO pemain (nama, point) VALUES ('%s',
%d)";
        sql=String.format(sql, nama, point);
        try {
            Class.forName(jdbc);
            koneksi = DriverManager.getConnection(url, user,
password);
            stat = koneksi.createStatement();

            statusTambah = stat.execute(sql);
            tutupDatabase();
        } catch (ClassNotFoundException | SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    public Database(String sql){ //untuk mengambil data
        try {
            Class.forName(jdbc);
            koneksi = DriverManager.getConnection(url, user,
password);
            stat = koneksi.createStatement();

            rs = stat.executeQuery(sql);
```

```
        } catch (ClassNotFoundException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE, null,
ex);
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    public boolean getStatusTambah(){
        return statusTambah;
    }

    public ResultSet getRS(){
        return rs;
    }

    public void tutupDatabase(){
        try {
            stat.close();
            koneksi.close();
        } catch (SQLException ex) {

Logger.getLogger(Database.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
}
```

Game.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package battleship;

import java.awt.Graphics;

/**
 *
 * @author ASUS R.O.G
 */
public abstract class Game {
    public abstract void gerak();
    public abstract void draw(Graphics g);
}
```