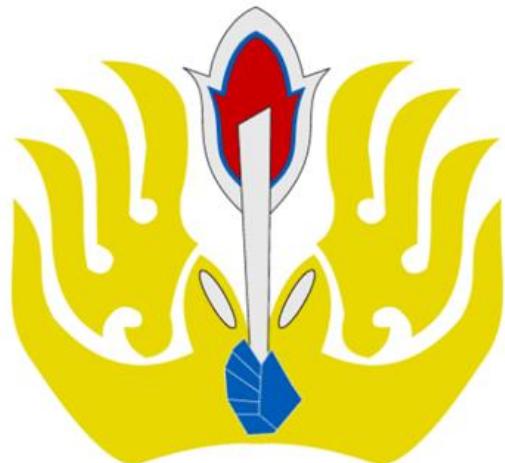


Laporan Projek UAS

Mata Kuliah Pemrograman Berorientasi Objek



Disusun oleh:

Muhammad Ali Akbar	2310631250094
Helmi Sulthan Muhammad	2310631250061
Abizard Zufar Fattahillah	2310631250001
Restu Nur Albar	2310631250027

Program Studi Sistem Informasi
Fakultas Ilmu Komputer
Universitas Singaperbangsa Karawang
Karawang
2025

DAFTAR ISI

Daftar Isi

DAFTAR ISI	2
BAB I PENDAHULUAN	4
1.1 Latar Belakang	4
1.2 Rumusan Masalah.....	5
1.3 Tujuan	5
BAB II PENGUJIAN DAN PENERAPAN	6
2.1 Fitur-Fitur Program Game Center	6
Login Multi-Role (Admin & Pengguna)	6
Registrasi Akun Pengguna.....	6
Dashboard Pengguna	6
Permainan Tic Tac Toe:	6
Permainan Snake Game:.....	7
Dashboard Admin	7
Koneksi Database MySQL	7
Utilitas Data Tabel.....	7
Validasi dan Penanganan Kesalahan.....	7
2.2 Konsep Object-Oriented Programming (OOP).....	8
Enkapsulasi (Encapsulation).....	8
Pewarisan (Inheritance)	8
Polimorfisme (Polymorphism)	9
Exception Handling	9
Modularisasi Kode Berbasis OOP	9
Penggunaan Event-Driven pada GUI	9
2.3 Unifield Modeling Language	9
2.3.1 USE CASE DIAGRAM	9
2.3.2 CLASS DIAGRAM.....	12
2.4 Pengujian dan penerapan.....	12
Admin.java	12
User.java.....	13
Score.java.....	15
DBConnection.java	17
DBUtils.java	19
Main.java	21

LoginFrame.java	22
DashboardFrame.java.....	27
AdminFrame.java	33
GameTicTacToe.java.....	44
GameUlar	50
Player2Login.java	65
RegisterFrame.java.....	81
BAB III PENUTUP.....	104
3.1 KESIMPULAN	104
3.2 SARAN	104

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang pesat telah mengubah cara kita berinteraksi dengan hiburan. Di era digital ini, game telah menjadi bentuk hiburan yang sangat populer, mendorong kebutuhan akan platform terpusat yang memungkinkan pengguna mengakses berbagai permainan, melacak skor, dan berinteraksi dalam lingkungan yang dinamis. Namun, seringkali pengguna menghadapi tantangan dalam menemukan satu tempat yang komprehensif untuk berbagai game kasual, pelacakan performa, dan fitur komunitas.

Melihat permasalahan tersebut, dibutuhkan suatu sistem GameCenter yang modern, fleksibel, dan dapat diakses dengan lebih efisien. Oleh karena itu, dikembangkanlah sebuah sistem GameCenter berbasis Java GUI (Graphical User Interface) yang terintegrasi dengan database MySQL. Sistem ini dirancang untuk mendukung proses digitalisasi pengalaman bermain game, mempermudah pengelolaan data oleh admin, serta memberikan kemudahan akses bagi pengguna dalam menjelajahi, bermain, dan melacak skor mereka. Sistem ini menggunakan pendekatan pemrograman berbasis objek (Object-Oriented Programming/OOP) yang mencakup penggunaan *class*, *inheritance*, *encapsulation*, *polymorphism*, serta *exception handling*, sehingga struktur kode menjadi modular, terstruktur, dan mudah dikembangkan lebih lanjut.

Dalam sistem ini, terdapat dua jenis pengguna utama yaitu admin dan pengguna umum (user). Admin memiliki akses untuk mengelola data pengguna terdaftar dan mengelola skor game yang ada. Di sisi lain, pengguna umum dapat melakukan registrasi dan login, melihat skor pribadi mereka, melihat papan peringkat global, serta memainkan game yang tersedia seperti Tic Tac Toe dan Snake Game. Fitur pelacakan skor dan papan peringkat meningkatkan aspek kompetitif dan rekam jejak pengguna.

Dari sisi antarmuka, sistem ini dirancang dengan tampilan modern dan responsif menggunakan komponen GUI berbasis Java Swing. Setiap tampilan menyesuaikan peran pengguna dan dirancang untuk mudah digunakan, bahkan oleh pengguna awam sekalipun. Proses login, registrasi, pemilihan game, dan pelacakan skor dilakukan dengan pendekatan *event-driven* yang intuitif. Dengan hadirnya sistem GameCenter ini, diharapkan pengelolaan data game dan pengguna menjadi lebih efisien, transparan, dan fleksibel. Pengguna dapat mengakses berbagai game tanpa batasan ruang dan waktu, sementara admin dapat mengelola data secara lebih akurat dan terorganisir. Sistem ini juga menjadi contoh penerapan nyata konsep pemrograman berbasis objek dan rekayasa perangkat lunak dalam menyelesaikan permasalahan nyata dalam bidang hiburan digital.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka rumusan masalah dari pengembangan sistem Game Center ini dapat dirumuskan sebagai berikut:

1. Bagaimana merancang dan mengimplementasikan sistem GameCenter berbasis Java GUI yang dapat memfasilitasi login, registrasi, dan permainan game secara efisien oleh pengguna?
2. Bagaimana cara mengatur hak akses yang berbeda antara admin dan pengguna umum dalam sistem GameCenter?
3. Bagaimana sistem dapat mengelola proses permainan, termasuk pelacakan skor dan kondisi menang/kalah/seri untuk game seperti Tic Tac Toe dan Snake Game?
4. Bagaimana merancang antarmuka sistem (GUI) yang interaktif, informatif, dan mudah digunakan bagi admin maupun pengguna?
5. Bagaimana mengintegrasikan database MySQL ke dalam sistem untuk menyimpan dan mengelola data pengguna, serta riwayat dan skor permainan secara terstruktur dan aman?

1.3 Tujuan

Adapun tujuan dari pembangunan sistem Game Center berbasis Java GUI ini adalah sebagai berikut:

1. Membangun sebuah sistem aplikasi GameCenter yang mampu mengelola registrasi dan autentikasi pengguna secara terpusat dan efisien.
2. Mengimplementasikan konsep pemrograman berbasis objek (OOP) dalam pengembangan sistem yang modular, terstruktur, dan mudah dikembangkan.
3. Memberikan fitur otentikasi login dengan pembagian hak akses antara admin dan pengguna biasa, agar setiap aktor hanya dapat mengakses fitur yang sesuai.
4. Memfasilitasi pengguna dalam proses pemilihan dan bermain game (Tic Tac Toe dan Snake Game) secara mandiri.
5. Memungkinkan pengguna melihat skor pribadi dan papan peringkat global untuk game yang dimainkan.
6. Menerapkan mekanisme penyimpanan dan pembaruan skor game ke database.
7. Membangun antarmuka sistem menggunakan Java Swing yang responsif, menarik, dan ramah pengguna (user-friendly).

BAB II

PENGUJIAN DAN PENERAPAN

2.1 Fitur-Fitur Program Game Center

Fitur-fitur yang tersedia dalam sistem GameCenter berbasis Java GUI + MySQL ini dirancang untuk mempermudah proses pengelolaan dan akses terhadap permainan secara digital. Sistem menyediakan dua jenis akses pengguna, yaitu Admin dan Pengguna Umum, dengan hak dan tampilan fitur yang berbeda. Berikut penjelasan lengkap setiap fitur:

Login Multi-Role (Admin & Pengguna)

Sistem menyediakan halaman login (LoginFrame.java) yang dapat digunakan oleh dua peran utama, yaitu admin dan pengguna umum. Pengguna dapat memasukkan *username* dan *password*. Jika autentikasi berhasil, sistem akan mengarahkan pengguna ke *dashboard* sesuai dengan *role*-nya. Proses login dilengkapi dengan validasi dan pesan kesalahan apabila data tidak sesuai. kesalahan apabila data tidak sesuai.

Registrasi Akun Pengguna

Pengguna baru dapat mendaftar melalui RegisterFrame.java dengan mengisi *username*, *password*, dan *nickname*. Sistem akan memvalidasi input dan memeriksa ketersediaan *username* di database sebelum menambahkan pengguna baru.

Dashboard Pengguna

Setelah login, pengguna diarahkan ke DashboardFrame.java. Dashboard ini menampilkan sapaan selamat datang, skor pribadi pengguna (misalnya, jumlah kemenangan Tic Tac Toe dan skor tertinggi Snake Game), serta papan peringkat global untuk kedua game tersebut. Tampilan papan peringkat menampilkan 5 pemain teratas.

Permainan Tic Tac Toe:

- Pengguna dapat memulai game Tic Tac Toe dari *dashboard*. Game ini dirancang untuk dua pemain, sehingga sebelum memulai, Player2LoginFrame.java akan meminta pemain kedua untuk login.
- Papan permainan berukuran 15x15 kotak. Pemain bergantian menempatkan simbol 'X' atau 'O'.
- Sistem mendeteksi pemenang ketika salah satu pemain berhasil menempatkan 5 simbol secara berurutan (horizontal, vertikal, atau diagonal).
- Setelah game berakhir (ada pemenang atau seri), skor (jumlah kemenangan) akan disimpan ke database untuk pemain yang menang.
- Terdapat tombol "Back to Dashboard" untuk kembali ke menu utama.

Permainan Snake Game:

- Pengguna dapat memulai game Snake Game dari *dashboard*. Game ini dimainkan secara solo.
- Papan permainan berukuran 20x20 kotak.
- Ular bergerak dan skor bertambah setiap kali ular memakan makanan.
- Game berakhir (*Game Over*) jika ular menabrak dinding atau tubuhnya sendiri.
- Skor tertinggi yang dicapai pengguna akan disimpan di database.
- Terdapat tombol "Back to Dashboard" untuk kembali ke menu utama.

Dashboard Admin

Admin memiliki akses ke AdminFrame.java. Di sini, admin dapat:

- Melihat daftar seluruh pengguna terdaftar dalam sistem pada tab "Users".
- Melakukan penghapusan pengguna terpilih dari daftar (yang juga akan menghapus skor terkait karena batasan kunci asing).
- Melihat daftar semua skor game pada tab "Scores".
- Melakukan pengeditan nilai skor yang dipilih.
- Melakukan penghapusan skor yang dipilih.
- Tersedia tombol "Refresh" untuk memuat ulang data pada kedua tab.
- Tersedia tombol "Logout" untuk kembali ke LoginFrame.java.

Koneksi Database MySQL

Sistem terhubung dengan database MySQL melalui koneksi JDBC. Kelas DBConnection.java digunakan untuk mengatur koneksi secara modular dan terpisah dari logika utama aplikasi.

Utilitas Data Tabel

Kelas DbUtils.java menyediakan metode statis untuk mengkonversi ResultSet dari query database menjadi TableModel yang cocok untuk JTable, mempermudah tampilan data di GUI.

Validasi dan Penanganan Kesalahan

Setiap form dalam sistem dilengkapi dengan validasi (misalnya, input wajib diisi, pengecekan kesesuaian login). Selain itu, sistem juga menerapkan try-catch (exception handling) untuk menghindari *crash* akibat *error* koneksi database atau input invalid.

2.2 Konsep Object-Oriented Programming (OOP)

Sistem GameCenter ini seluruh strukturnya dibangun berdasarkan prinsip-prinsip Pemrograman Berbasis Objek (Object-Oriented Programming/OOP) yang meliputi: *Class*, *Object*, Enkapsulasi, Pewarisan (*Inheritance*) (implisit melalui konsep admin sebagai jenis user), Polimorfisme (*Polymorphism*), dan *Exception Handling*. Berikut penjelasan rinci mengenai penerapan konsep tersebut:

Class dan Object *Class* digunakan sebagai *blueprint* atau cetakan untuk membuat objek. Dalam sistem ini, setiap entitas penting diwakili oleh sebuah *class*, seperti:

- User.java → merepresentasikan data pengguna secara umum.
- Score.java → merepresentasikan data skor permainan.
- AdminFrame.java → merepresentasikan antarmuka admin.
- DashboardFrame.java → merepresentasikan antarmuka *dashboard* pengguna.
- DBConnection.java → merepresentasikan modul koneksi database.
- DbUtils.java → merepresentasikan utilitas bantu untuk database dan tabel.
- GameTicTacToe.java → merepresentasikan logika permainan Tic Tac Toe.
- GameUlar.java → merepresentasikan logika permainan Snake Game.
- LoginFrame.java → merepresentasikan antarmuka login pengguna.
- Main.java → merepresentasikan titik masuk utama aplikasi.
- Player2LoginFrame.java → merepresentasikan antarmuka login pemain kedua.
- RegisterFrame.java → merepresentasikan antarmuka registrasi akun baru.

Contoh: `User adminUser = new User(1, "admin", "password123", "Admin");` Kode di atas membuat objek User dengan data lengkap sebagai *instance* dari *class* User.

Enkapsulasi (Encapsulation)

Enkapsulasi diterapkan dengan cara menjadikan semua atribut dalam class bersifat private dan hanya dapat diakses melalui *getter* dan *setter*. Tujuannya untuk menjaga integritas data dan mencegah akses langsung dari luar *class*. Contoh pada User.java: `private String username; public String getUsername() { return username; }`.

Pewarisan (Inheritance)

Konsep *inheritance* diterapkan pada Admin.java yang extends User.java. Ini berarti kelas Admin mewarisi semua atribut dan metode dari User, secara eksplisit menyatakan hubungan "adalah seorang" (An Admin is a User). Contoh implementasi dapat dilihat pada deklarasi `public class Admin extends User { ... }` atau DashboardFrame.

Polimorfisme (Polymorphism)

Polimorfisme memungkinkan objek Admin diperlakukan sebagai objek User karena Admin adalah turunan dari User. Ini terlihat dalam LoginFrame.java di mana objek yang dibuat (User atau Admin) diteruskan ke DashboardFrame.java yang menerima parameter User. Meskipun AdminFrame.java secara spesifik mengharapkan objek Admin, DashboardFrame.java dapat secara fleksibel menangani keduanya jika diperlukan.

Exception Handling

Penggunaan try-catch digunakan untuk menangani kesalahan program, seperti kegagalan koneksi database, kesalahan login, atau input invalid. Contoh: try (Connection conn = DBConnection.getConnection() { // ... } catch (SQLException e) { JOptionPane.showMessageDialog(this, "Error koneksi database: " + e.getMessage()); }. Ini melindungi sistem dari *crash* saat terjadi *error*.

Modularisasi Kode Berbasis OOP

Dengan OOP, kode dibagi dalam *package* sesuai tanggung jawab:

- code/: Berisi kelas-kelas utama UI dan logika aplikasi (misalnya, LoginFrame, DashboardFrame, GameTicTacToe, GameUlar, Main).
- code/model/: Berisi kelas-kelas entitas data (User, Score, Admin).
- Setiap komponen dipisahkan agar lebih mudah dipelihara dan dikembangkan (prinsip Single Responsibility).

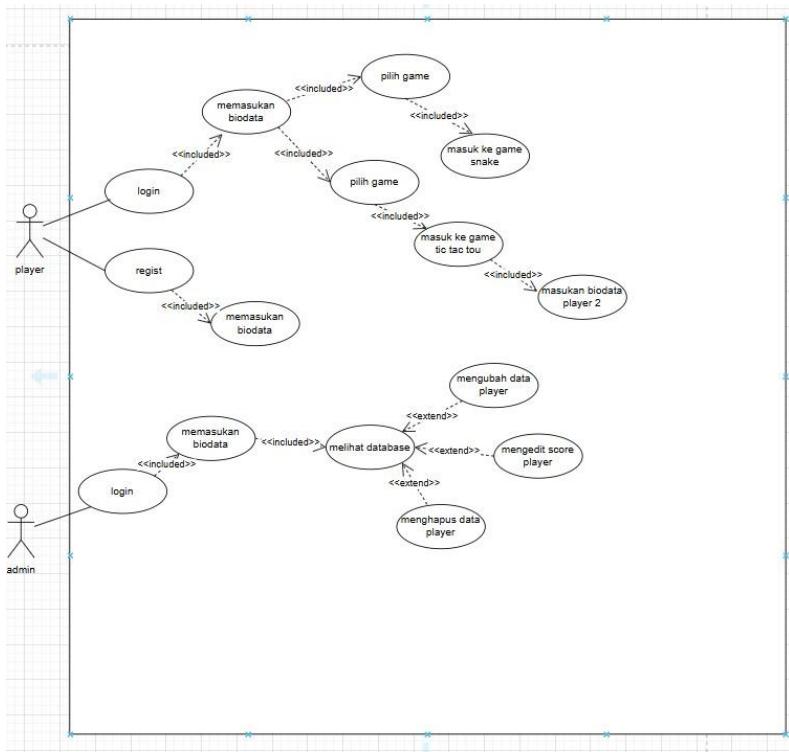
Penggunaan Event-Driven pada GUI

Dalam kelas GUI seperti LoginFrame dan DashboardFrame, digunakan *action listener* sebagai implementasi prinsip *event-driven programming*. Artinya, sistem menunggu aksi pengguna (klik tombol, input, dll) untuk mengeksekusi method tertentu. Contoh: logoutButton.addActionListener(e -> { new LoginFrame().setVisible(true); dispose(); }).

2.3 Unifield Modeling Language

2.3.1 USE CASE DIAGRAM

Use case diagram adalah salah satu jenis diagram UML (Unified Modeling Language) yang digunakan untuk memodelkan interaksi antara pengguna (aktor) dengan sistem berdasarkan fungsionalitas yang disediakan. Diagram ini menggambarkan siapa saja yang menggunakan sistem dan fitur-fitur apa saja yang dapat mereka akses.



Jika dilihat pada gambar use case diagram di atas, dapat diketahui bahwa terdapat dua aktor utama yang terlibat dalam sistem, yaitu Player dan Admin. Masing-masing aktor memiliki hak akses dan aktivitas yang berbeda-beda di dalam sistem sesuai dengan perannya.

Seorang Player dapat melakukan beberapa kegiatan di dalam sistem. Pertama, register (regist), yaitu proses mendaftar sebagai pengguna baru. Proses ini hanya dilakukan jika seorang Player belum memiliki akun sebelumnya. Setelah berhasil melakukan registrasi, Player akan diminta untuk memasukkan biodata, yang merupakan bagian penting dan tidak bisa dipisahkan dari proses pendaftaran. Ini ditunjukkan dengan adanya relasi <<include>> antara use case regist dan memasukkan biodata. Artinya, kegiatan “memasukkan biodata” harus dilakukan saat melakukan pendaftaran.

Jika seorang Player sudah memiliki akun, maka mereka bisa langsung melakukan login. Saat login, Player juga harus memasukkan biodata (termasuk username dan password), karena ada relasi <<include>> antara login dan memasukkan biodata. Include di sini berarti bahwa suatu aktivitas selalu menyertakan aktivitas lainnya. Jadi, bila Player tidak mengisi biodata dengan benar, proses login akan gagal dan mereka tidak akan bisa mengakses sistem.

Setelah berhasil login, Player diarahkan untuk melakukan pemilihan game melalui use case pilih game. Kegiatan ini juga memiliki relasi <<include>> ke beberapa aktivitas, yaitu:

- Masuk ke game Snake

- Masuk ke game Tic Tac Toe

Artinya, saat seorang Player memilih game, mereka harus memilih salah satu dari dua jenis permainan yang tersedia.

Untuk game Tic Tac Toe, sistem mengharuskan pengguna untuk memasukkan biodata Player 2. Ini juga merupakan relasi <<include>>, yang menunjukkan bahwa pemain kedua perlu dikenali sebelum permainan dimulai. Jika data Player 2 belum diisi, maka game tidak bisa dilanjutkan.

◊ Admin

Admin merupakan aktor dengan hak akses lebih tinggi dalam sistem. Admin dapat melakukan login, dan seperti Player, mereka juga harus memasukkan biodata agar dapat masuk ke dalam sistem (juga ditunjukkan dengan <<include>>).

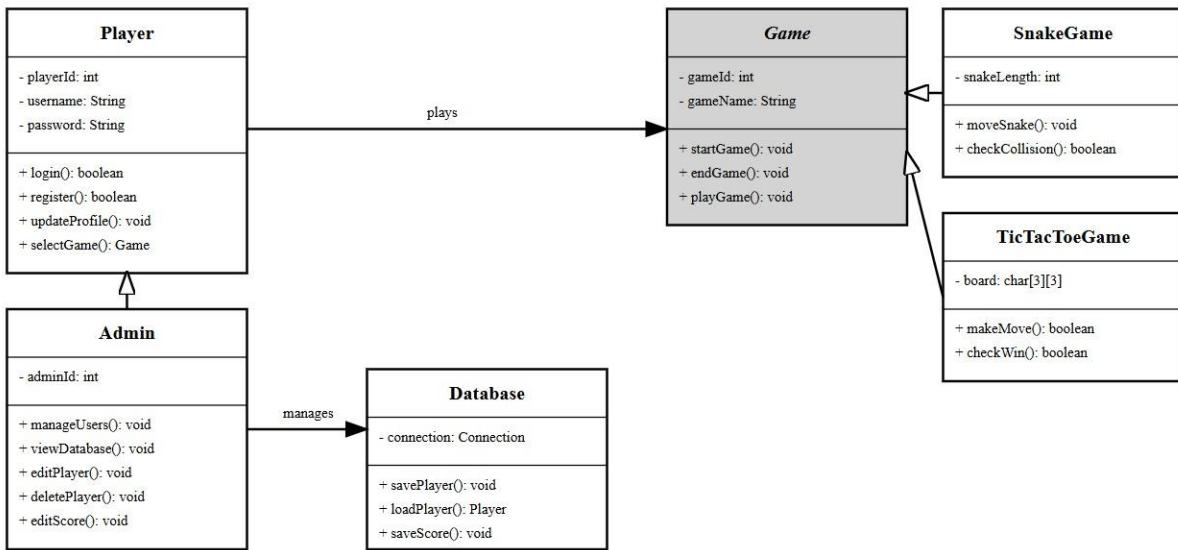
Setelah berhasil login, admin diarahkan ke menu melihat database. Dalam menu ini, admin dapat melihat seluruh data Player yang telah melakukan registrasi dan bermain game.

Dari use case melihat database, terdapat beberapa relasi <<extend>> ke fitur tambahan yang dapat dilakukan oleh admin, yaitu:

- Mengubah data player
- Mengedit score player
- Menghapus data player

Relasi <<extend>> di sini menunjukkan bahwa ketika admin melakukan kegiatan utama melihat database, mereka juga berkesempatan untuk memperluas aktivitasnya ke berbagai tindakan administratif lainnya. Namun, tindakan-tindakan ini bersifat opsional, tergantung dari kebutuhan dan kebijakan sistem. Misalnya, admin tidak selalu harus mengedit skor pemain, namun bisa melakukan itu jika diperlukan.

2.3.2 CLASS DIAGRAM



2.4 Pengujian dan penerapan

Admin.java

```

1 package code.model;
2
3 public class Admin extends User {
4     public Admin(int userId, String username, String password, String nickname) {
5         super(userId, username, password, nickname);
6     }
7 }
  
```

Line 1: Deklarasi Package: package code.model;

- Line ini mendeklarasikan bahwa file Java ini adalah bagian dari package code.model. Ini membantu dalam mengorganisir kode dan mencegah konflik penamaan kelas.

Line 3: Deklarasi Kelas Admin: public class Admin extends User {

- Line ini mendeklarasikan kelas publik bernama Admin. Kata kunci extends User menunjukkan bahwa Admin adalah *subclass* dari kelas User. Ini berarti kelas Admin akan mewarisi semua properti (variabel) dan perilaku (metode) dari kelas User, dan dapat menambahkan properti atau perilaku spesifik untuk seorang admin. Ini adalah contoh *inheritance* (pewarisan) dalam OOP.

Line 4: Konstruktor Admin: public Admin(int userId, String username, String password, String nickname) {

- Ini adalah konstruktor untuk kelas Admin. Ketika objek Admin baru dibuat, konstruktor ini akan dipanggil. Ia menerima empat parameter: userId (tipe integer), username (tipe String), password (tipe String), dan nickname (tipe String).

Line 5: Memanggil Konstruktor Superclass: super(userId, username, password, nickname);

- Line ini memanggil konstruktor dari *superclass* (User). Ini adalah cara untuk menginisialisasi bagian dari objek Admin yang diwarisi dari User. Ini meneruskan semua parameter yang diterima oleh konstruktor Admin langsung ke konstruktor User.

User.java

```
package code.model;

public class User {
    private int userId;
    private String username;
    private String password;
    private String nickname;

    public User(int userId, String username, String password, String nickname) {
        this.userId = userId;
        this.username = username;
        this.password = password;
        this.nickname = nickname;
    }
}
```

Penjelasan code

Line 1

- Menyatakan bahwa file ini berada di dalam package code.model, yaitu struktur folder yang digunakan untuk mengelompokkan kelas-kelas Java agar lebih terorganisir.

line 3-4

- Mendeklarasikan kelas user sebagai kelas publik yang dapat diakses dari kelas lain.
- Kelas ini merepresentasikan data pengguna (user) dalam bentuk objek. Merupakan atribut atau properti milik objek User.

Line 5-8

Semua atribut bersifat private, artinya hanya dapat diakses dari dalam kelas ini sendiri.

- userId: ID unik pengguna (integer).
- username: Nama pengguna.
- password: Kata sandi pengguna.
- nickname: Nama panggilan pengguna.

Line 10-15

- Merupakan **constructor** untuk membuat objek User.
- Parameter yang diterima akan digunakan untuk mengisi nilai dari atribut objek.
- Keyword this digunakan untuk membedakan antara variabel instance dan parameter.

```
// Getter methods
public int getUserId() { return userId; }
public String getUsername() { return username; }
public String getPassword() { return password; }
public String getNickname() { return nickname; }

// Setter methods
public void setPassword(String password) { this.password = password; }
public void setNickname(String nickname) { this.nickname = nickname; }
```

Line 18-21 (Getter Methods)

- Method getter digunakan untuk mengambil (mengakses) nilai dari atribut yang bersifat private.
- Dapat dipanggil dari luar kelas untuk mendapatkan data user.

Line 24-25 (Setter Methods)

- Method setter digunakan untuk mengubah nilai dari atribut password dan nickname.
- Tidak tersedia setter untuk userId dan username, yang berarti nilainya tidak bisa diubah setelah objek dibuat.

Score.java

```
1 package code.model;
2 import java.sql.Timestamp;
3
4 public class Score {
5     private int scoreId;
6     private int userId;
7     private String gameName;
8     private int score;
9     private Timestamp lastPlayed;
```

Penjelasan code

Line 1-9

Baris 1: Menyatakan bahwa file ini merupakan bagian dari package code.model, yang biasanya digunakan untuk mengorganisir kelas-kelas dalam struktur proyek Java.

Baris 2: Mengimpor class Timestamp dari java.sql, yang digunakan untuk menyimpan informasi waktu dan tanggal saat skor terakhir dimainkan.

Baris 4-9: Deklarasi kelas Score yang merepresentasikan data skor pemain.

Di dalamnya terdapat atribut:

- scoreId: ID unik untuk data skor.
- userId: ID pengguna yang memiliki skor ini.
- gameName: Nama game yang dimainkan.
- score: Nilai skor yang diperoleh pengguna.
- lastPlayed: Waktu terakhir game tersebut dimainkan, bertipe Times

```
11     public Score(int scoreId, int userId, String gameName, int score, Timestamp lastPlayed) {
12         this.scoreId = scoreId;
13         this.userId = userId;
14         this.gameName = gameName;
15         this.score = score;
16         this.lastPlayed = lastPlayed;
17     }
```

Line 11 - 17

- Baris ini adalah konstruktor dari kelas Score.
- Konstruktor digunakan untuk menginisialisasi objek Score dengan nilai-nilai awal ketika objek dibuat.
- Kata kunci this digunakan untuk membedakan antara atribut kelas dan parameter konstruktor.
- Parameter yang diberikan ke konstruktor akan disimpan ke masing-masing atribut objek.

```
18 ...
19 ... // Getter methods
20 ... public int getScoreId() { return scoreId; }
21 ... public int getUserId() { return userId; }
22 ... public String getGameName() { return gameName; }
23 ... public int getScore() { return score; }
24 ... public Timestamp getLastPlayed() { return lastPlayed; }
25 }
```

Line 19-25

Baris-baris ini adalah method getter yang digunakan untuk mengambil nilai dari atribut yang bersifat private.

Setiap method akan mengembalikan nilai dari atribut tertentu:

- `getScoreId()`: Mengembalikan nilai `scoreId`.
- `getUserId()`: Mengembalikan nilai `userId`.
- `getGameName()`: Mengembalikan nama game.
- `getScore()`: Mengembalikan skor.
- `getLastPlayed()`: Mengembalikan waktu terakhir game dimainkan.

DBConnection.java

```
1 package code;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DBConnection {
8     ... private static final String URL = "jdbc:mysql://localhost:3306/gudang_game?useSSL=false&serverTimezone=UTC";
9     ... private static final String USER = "root";
10    ... private static final String PASSWORD = "";
```

Penjelasan code

- Line 1:
Menyatakan bahwa file ini berada dalam package code, yang digunakan untuk mengelompokkan kelas-kelas Java berdasarkan struktur modular.
- Line 3–5:
Mengimpor library JDBC dari Java yang diperlukan untuk membuat koneksi dengan database:
 - a. java.sql.Connection: Digunakan untuk merepresentasikan koneksi ke database.
 - b. java.sql.DriverManager: Mengelola driver database dan membuat koneksi.
 - c. java.sql.SQLException: Menangani kesalahan SQL yang mungkin terjadi saat berinteraksi dengan database.
- Line 7:
Deklarasi kelas DBConnection sebagai kelas publik, yang akan digunakan untuk menangani koneksi ke database.
- Line 8–10:
Mendefinisikan tiga konstanta (final) statis yang menyimpan informasi koneksi database:
 - a. URL: Alamat JDBC dari database MySQL lokal dengan nama database gudang_game, serta parameter tambahan useSSL=false dan serverTimezone=UTC.
 - b. USER: Nama pengguna database (dalam hal ini root).
 - c. PASSWORD: Kata sandi pengguna (kosong dalam contoh ini).

```
12     public static Connection getConnection() throws SQLException {  
13         try {  
14             Class.forName(className:"com.mysql.cj.jdbc.Driver");  
15             return DriverManager.getConnection(URL, USER, PASSWORD);  
16         } catch (ClassNotFoundException e) {  
17             throw new SQLException(reason:"MySQL JDBC Driver not found", e);  
18         }  
19     }
```

Line 12:

- Mendeklarasikan method statik getConnection() yang mengembalikan objek Connection dan dapat melempar SQLException.

Line 13–14 (try block):

- Mencoba untuk memuat driver MySQL dengan Class.forName("com.mysql.cj.jdbc.Driver"). Ini diperlukan untuk menginisialisasi driver JDBC.

Line 15:

- Jika driver berhasil dimuat, maka method akan mencoba membuka koneksi ke database menggunakan DriverManager.getConnection(URL, USER, PASSWORD).

Line 16–17 (catch block):

- Jika driver MySQL tidak ditemukan (menyebabkan ClassNotFoundException), maka blok catch akan menangkapnya dan melempar SQLException baru dengan pesan: "MySQL JDBC Driver not found", beserta penyebab kesalahan (e).

Line 19:

- Akhir dari method getConnection.

DBUtils.java

```
1 package code;
2
3 import javax.swing.table.*;
4 import java.sql.*;
5 import java.util.ArrayList;
6 import java.util.List;
```

Penjelasan code

Line 1 - 6

- Bagian ini adalah persiapan. Kode ini mengimpor semua pustaka (library) yang dibutuhkan, terutama untuk koneksi database (java.sql), pembuatan tabel di antarmuka (javax.swing.table), dan struktur data (java.util.List).

```
8 public class Dbutils {
9     ...
10    public static TableModel resultSetToTableModel(ResultSet rs) throws SQLException {
11        ...
12        ...
13        ...
14        ...
15        ...
16        ...
17        ...
18        ...
19        ...
20        ...
21        ...
22        ...
23        ...
24        ...
25        ...
26        ...
27        ...
28 }
```

Line 8 - 27

- Bagian ini adalah inti proses pengambilan data. Kode ini mendefinisikan sebuah fungsi yang pertama-tama mengambil semua **nama kolom** dari hasil kueri database, lalu berlanjut membaca **setiap baris data** satu per satu dan menyimpannya sementara ke dalam sebuah daftar (List).

```
29     .....// Convert list to array
30     .....Object[][] data = new Object[dataList.size()][];
31     .....for (int i = 0; i < dataList.size(); i++) {
32     .....    data[i] = dataList.get(i);
33     ..}
34
35     .....return new DefaultTableModel(data, columnNames) {
36     .....    @Override
37     .....        public boolean isCellEditable(int row, int column) {
38     .....            .....return false;
39     .....        }
40     ..};
41
42 }
```

Line 29 - 42

- Bagian ini adalah tahap akhir. Data yang sudah terkumpul diubah menjadi format TableModel yang siap pakai. Kemudian, model tabel ini dikembalikan sebagai hasil akhir dan diatur agar tidak bisa diubah (non-editable) oleh pengguna.

Main.java



```
1 package code;
2
3 public class Main {
4     public static void main(String[] args) {
5         // Initialize database connection
6         try {
7             DBConnection.getConnection();
8             System.out.println("DATABASE TERHUBUNG");
9         } catch (Exception e) {
10             System.err.println("GAGAL MENYAMBUNG DATABASE: " + e.getMessage());
11             return;
12         }
13
14         // Start the application
15         new LoginFrame().setVisible(true);
16     }
17 }
```

penjelasan code

line 1-5

- Menyatakan bahwa file ini termasuk dalam package code, yaitu folder yang mengelompokkan kelas-kelas Java.
- Kelas Main adalah kelas utama dalam program ini dan memiliki method main, yaitu titik awal (entry point) ketika aplikasi dijalankan. Semua proses eksekusi program akan dimulai dari method ini.

line 6-12

- Blok try ini mencoba untuk melakukan koneksi ke database menggunakan method DBConnection.getConnection().
Jika koneksi berhasil, maka akan menampilkan pesan "DATABASE TERHUBUNG" ke konsol sebagai konfirmasi.
- Blok catch akan menangkap jika terjadi error saat koneksi ke database.
- Akan ditampilkan pesan error: "GAGAL MENYAMBUNG DATABASE" beserta rincian kesalahan, kemudian program akan dihentikan dengan return.

line 13-17

- Blok catch menangani kemungkinan gagalnya koneksi ke database.
Jika terjadi error, maka akan ditampilkan pesan kesalahan ke konsol, dan program akan dihentikan dengan perintah return.
- Membuat dan menampilkan form login (LoginFrame) sebagai tampilan pertama yang akan dilihat oleh pengguna.

output program ini

```
PS C:\Users\helmi\OneDrive\Desktop\uaspbo> c:; cd 'c:\Users\helmi\OneDrive\Desktop\uaspbo'; & 'C:\Program Files\Java\jdk-22\bin\java.exe' '@C:\Users\helmi\AppData\Local\Temp\cp_e7t2ntjcop09jr9yka0ces3h2.argfile' 'code.Main'
DATABASE TERHUBUNG
```

LoginFrame.java

Penjelasan Code

```
1 package code;
2
3 import code.model.Admin; // Import kelas Admin
```

line 1-3 Package dan import

- Menentukan bahwa kelas ini berada dalam *package* code.
- Mengimpor kelas Admin dan User dari sub-package code.model.

```
4 import code.model.User;
5 import javax.swing.*;
6 import java.awt.*;
7 import java.awt.event.ActionEvent;
8 import java.sql.*;
```

line 4-8 Swing dan AWT

- Mengimpor kelas dari javax.swing untuk GUI.
- java.awt dan java.awt.event untuk layout dan event handling.
- java.sql digunakan untuk operasi basis data

```
10 public class LoginFrame extends JFrame {
11     private JTextField usernameField;
12     private JPasswordField passwordField;
```

line 9-12 Deklarasi kelas dan variabel

- LoginFrame adalah turunan dari JFrame, yaitu jendela utama GUI.
- usernameField dan passwordField adalah komponen input teks.

```
12     private JPasswordField passwordField;
13
14     public LoginFrame() {
15         setTitle("Game Center Login");
16         setSize(900, 550);
17         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18         setLocationRelativeTo(null);
19     }
```

line 12-18 constructor dan setup frame

- Menetapkan judul, ukuran frame, aksi saat ditutup (exit app), dan lokasi jendela di tengah layar.

```
20         getContentPane().setBackground(new Color(30, 30, 30));
21         setLayout(new BorderLayout());
```

line 20-21 Pengaturan warna dan layout

- Mengatur latar belakang hitam keabu-abuan.
- Menggunakan layout BorderLayout untuk tata letak komponen.

```
22
23     JPanel contentPanel = new JPanel();
24     contentPanel.setLayout(new GridBagLayout());
25     contentPanel.setBackground(new Color(30, 30, 30));
26
```

line 22-26 Panel utama (content panel)

- Membuat panel tengah dengan layout GridBagLayout agar lebih fleksibel dan rapi.

```
26
27     GridBagConstraints gbc = new GridBagConstraints();
28     gbc.insets = new Insets(10, 0, 10, 0);
29     gbc.fill = GridBagConstraints.HORIZONTAL;
30     gbc.anchor = GridBagConstraints.CENTER;
31
```

line 27-31 GridBagConstraints untuk pengaturan posisi komponen

- gbc digunakan untuk mengatur margin dan posisi setiap komponen di GridBagLayout.

```
32     JLabel titleLabel = new JLabel("Game Center");
33     titleLabel.setFont(new Font("Arial", Font.BOLD, 36));
34     titleLabel.setForeground(Color.WHITE);
35     gbc.gridx = 0;
36     gbc.gridy = 0;
37     gbc.gridwidth = 2;
38     gbc.insets = new Insets(0, 0, 40, 0);
39     contentPanel.add(titleLabel, gbc);
40
```

line 32-40 Label judul

- Menampilkan label judul besar di atas form login.

```
43     JLabel usernameLabel = new JLabel("Username");
44     usernameLabel.setFont(new Font("Arial", Font.PLAIN, 16));
45     usernameLabel.setForeground(Color.WHITE);
46     gbc.gridx = 0;
47     gbc.gridy = 1;
48     gbc.anchor = GridBagConstraints.WEST;
49     contentPanel.add(usernameLabel, gbc);
50
51
```

line 43-51 Reset gridwidth dan insets

- Menyesuaikan ulang grid sebelum menambahkan input.

```
51     usernameField = new JTextField(25);
52     usernameField.setFont(new Font("Arial", Font.PLAIN, 18));
53     usernameField.setBackground(Color.WHITE);
54     usernameField.setForeground(Color.BLACK);
55     usernameField.setBorder(BorderFactory.createCompoundBorder(
56         BorderFactory.createLineBorder(new Color(200, 200, 200), 1),
57         BorderFactory.createEmptyBorder(10, 15, 10, 15)
58     ));
59     gbc.gridx = 0;
60     gbc.gridy = 2;
61     gbc.insets = new Insets(0, 0, 20, 0);
62     gbc.anchor = GridBagConstraints.CENTER;
63     contentPanel.add(usernameField, gbc);
64
65
```

line 51-65 kolom input dan menempelkan username

- Membuat kolom input teks untuk username dengan style custom.
- Menempatkan kolom username di posisi tertentu di grid.

```

65
66     JLabel passwordLabel = new JLabel("Password");
67     passwordLabel.setFont(new Font("Arial", Font.PLAIN, 16));
68     passwordLabel.setForeground(Color.WHITE);
69     gbc.gridx = 0;
70     gbc.gridy = 3;
71     gbc.insets = new Insets(10, 0, 5, 0);
72     gbc.anchor = GridBagConstraints.WEST;
73     contentPanel.add(passwordLabel, gbc);

74
75     passwordField = new JPasswordField(25);
76     passwordField.setFont(new Font("Arial", Font.PLAIN, 18));
77     passwordField.setBackground(Color.WHITE);
78     passwordField.setForeground(Color.BLACK);
79     passwordField.setBorderStyle(BorderFactory.createCompoundBorder(
80         BorderFactory.createLineBorder(new Color(200, 200, 200), 1),
81         BorderFactory.createEmptyBorder(10, 15, 10, 15)
82     ));
83     gbc.gridx = 0;
84     gbc.gridy = 4;
85     gbc.insets = new Insets(0, 0, 30, 0);
86     gbc.anchor = GridBagConstraints.CENTER;
87     contentPanel.add(passwordField, gbc);
88

```

line 66-88 label password dan input password field

- Label untuk input password, mirip dengan username.
- Input field untuk password, disembunyikan (*hidden characters*).
- Menempatkan kolom password di bawah kolom username.

```

88
89     JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER,
90     buttonPanel.setBackground(new Color(30, 30, 30));
91
92     JButton loginButton = new JButton("Login");
93     loginButton.setFont(new Font("Arial", Font.BOLD, 18));
94     loginButton.setBackground(new Color(255, 165, 0));
95     loginButton.setForeground(Color.WHITE);
96     loginButton.setPreferredSize(new Dimension(150, 45));
97     loginButton.setFocusPainted(false);
98     loginButton.setBorderPainted(false);
99     buttonPanel.add(loginButton);
100    loginButton.addActionListener(this::performLogin);
101
102    JButton registerButton = new JButton("Register");
103    registerButton.setFont(new Font("Arial", Font.BOLD, 18));
104    registerButton.setBackground(new Color(76, 175, 80));
105    registerButton.setForeground(Color.WHITE);
106    registerButton.setPreferredSize(new Dimension(150, 45));
107    registerButton.setFocusPainted(false);
108    registerButton.setBorderPainted(false);
109    buttonPanel.add(registerButton);
110    registerButton.addActionListener(e -> {
111        new RegisterFrame().setVisible(true);
112        dispose();
113    });
114

```

line 88-113 panel tombol login dan register

- Membuat panel horizontal untuk tombol-tombol.
- Tombol login akan memicu method performLogin.
- Tombol register akan membuka RegisterFrame dan menutup frame ini.

```

114
115     gbc.gridx = 0;
116     gbc.gridy = 5;
117     gbc.gridwidth = 2;
118     gbc.insets = new Insets(0, 0, 0, 0);
119     contentPanel.add(buttonPanel, gbc);
120
121     add(contentPanel, BorderLayout.CENTER);
122 }
123

```

line 114-123 Menempatkan dan menambahkan panel tombol ke content panel

- Meletakkan tombol login dan register di bawah input.
- Menambahkan contentPanel ke tengah jendela utama.

```

124     private void performLogin(ActionEvent e) {
125         String username = usernameField.getText().trim();
126         String password = new String(passwordField.getPassword()).trim();
127
128         if (username.isEmpty() || password.isEmpty()) {
129             JOptionPane.showMessageDialog(this, "Username dan password harus diisi");
130             return;
131         }
132
133     try (Connection conn = DBConnection.getConnection()) { ...
134     } catch (SQLException ex) {
135         JOptionPane.showMessageDialog(this, "Error koneksi database: " + ex.getMessage());
136         ex.printStackTrace();
137     }
138 }
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171

```

line 124-165 Method performLogin

- Mengecek apakah input username & password kosong.
- Melakukan koneksi ke database melalui DBConnection.getConnection().
- Menggunakan SQL untuk mengecek user: "SELECT user_id, username, nickname, is_admin FROM users WHERE username = ? AND password = ?"

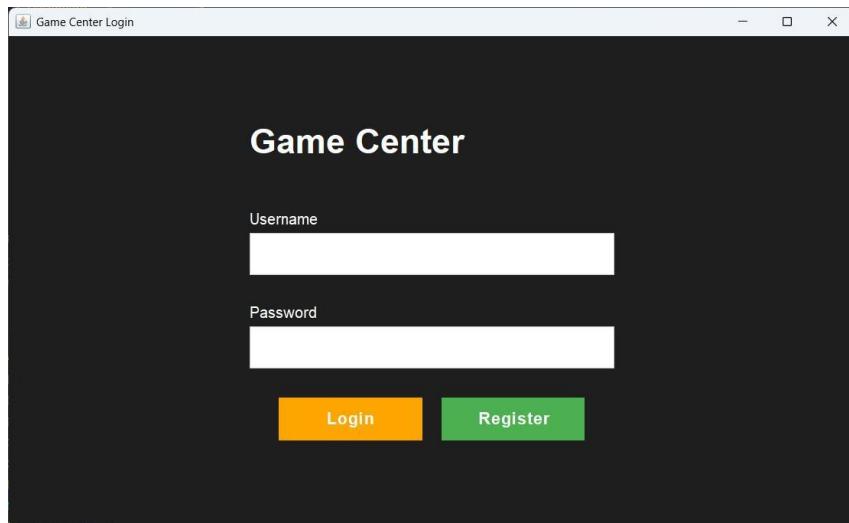
```

164
165
166     Run main | Debug main
167     public static void main(String[] args) {
168         SwingUtilities.invokeLater(() -> {
169             new LoginFrame().setVisible(true);
170         });
171     }

```

line 166-171 main method

- Titik awal eksekusi program. Membuka GUI login pada thread GUI (EDT).



Output

Ketika file dijalankan, maka akan keluar tampilan form login seperti ini. Player diminta untuk mengisi form login jika sudah memiliki akun, jika player belum memiliki akun customer bisa klik button register.

DashboardFrame.java

penjelasan code

```

1 package code;
2
3 import code.model.User;
4 import javax.swing.*;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.sql.*;

8
9 public class DashboardFrame extends JFrame {
10     private final User user;
11     private JTextArea userScoresArea;
12     private JTextArea leaderboardArea;

```

line 1-12 Import dan Deklarasi Kelas

- Line 1: Menentukan bahwa kelas ini berada dalam package code.
- Line 3: Mengimpor kelas User dari package code.model.
- Line 4-6: Mengimpor pustaka GUI Swing, dan pustaka AWT untuk manipulasi komponen dan warna.
- Line 7: Mengimpor pustaka java.sql untuk koneksi ke database.

- Line 8: Mendeklarasikan kelas DashboardFrame yang merupakan turunan dari JFrame, artinya ini adalah jendela utama tampilan GUI.
- Line 9: Mendeklarasikan variabel user yang menyimpan data pengguna yang sedang login.
- Line 10-11: Mendeklarasikan area teks untuk menampilkan skor pengguna dan leaderboard.

```

13
14     // Konstruktor menerima objek User, bisa Admin atau User biasa
15     public DashboardFrame(User user) {
16         this.user = user;
17         setTitle("Game Center - Welcome " + user.getNickname());
18         setSize(900, 550);
19         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         setLocationRelativeTo(null);
21
22         getContentPane().setBackground(new Color(30, 30, 30));
23

```

line 13-22 Konstruktor GUI

- Line 15: Konstruktor menerima parameter User yang sudah login.
- Line 16: Menyimpan data user ke variabel kelas.
- Line 17: Mengatur judul jendela berdasarkan nama pengguna.
- Line 18: Mengatur ukuran jendela ke 900x550 piksel.
- Line 19: Menutup aplikasi saat jendela ditutup.
- Line 20: Menampilkan jendela di tengah layar.
- Line 22: Mengatur warna latar belakang jendela menjadi gelap.

```

23
24     JPanel mainPanel = new JPanel(new BorderLayout(15, 15));
25     mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
26     mainPanel.setBackground(new Color(30, 30, 30));
27
28     JPanel topPanel = new JPanel(new BorderLayout());
29     topPanel.setBackground(new Color(30, 30, 30));
30
31     JLabel welcomeLabel = new JLabel("Welcome, " + user.getNickname() + "!", JLabel.CENTER);
32     welcomeLabel.setFont(new Font("Arial", Font.BOLD, 24));
33     welcomeLabel.setForeground(Color.WHITE);
34     topPanel.add(welcomeLabel, BorderLayout.CENTER);
35
36     JButton logoutButton = new JButton("Logout");
37     logoutButton.setFont(new Font("Arial", Font.BOLD, 14));
38     logoutButton.setBackground(new Color(220, 53, 69));
39     logoutButton.setForeground(Color.WHITE);
40     logoutButton.setFocusPainted(false);
41     logoutButton.setBorderPainted(false);
42     logoutButton.setPreferredSize(new Dimension(100, 35));
43 >     logoutButton.addActionListener(e -> { ...
44     JPanel logoutButtonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 0, 0));
45     logoutButtonPanel.setBackground(new Color(30, 30, 30));
46     logoutButtonPanel.add(logoutButton);
47     topPanel.add(logoutButtonPanel, BorderLayout.EAST);
48
49     mainPanel.add(topPanel, BorderLayout.NORTH);
50
51
52
53

```

line 24-53 Panel Utama dan Panel Atas (Header)

- Line 24-26: Membuat panel utama dengan layout border dan padding. Latar belakangnya gelap.
- Line 27-29: Panel atas untuk header, berisi tulisan sambutan dan tombol logout.
- Line 31-35: Label sambutan pengguna dengan ukuran font besar, diletakkan di tengah atas panel.
- Line 36-42: Membuat tombol logout dengan gaya merah khas tombol keluar.
- Line 43-46: Saat tombol ditekan, kembali ke LoginFrame dan menutup dashboard.
- Line 47-53: Menambahkan tombol logout ke pojok kanan atas.



```
1 JPanel centerPanel = new JPanel(new GridLayout(1, 2, 20, 10));
2 centerPanel.setBackground(new Color(30, 30, 30));
3
4 JPanel userScoresPanel = new JPanel(new BorderLayout());
5 userScoresPanel.setBackground(new Color(45, 45, 45));
6 userScoresPanel.setBorder(BorderFactory.createTitledBorder(
7     BorderFactory.createLineBorder(new Color(75, 75, 75)),
8     "Your Scores",
9     javax.swing.border.TitledBorder.LEFT,
10    javax.swing.border.TitledBorder.TOP,
11    new Font("Arial", Font.BOLD, 16),
12    Color.WHITE
13));
14 userScoresArea = new JTextArea();
15 userScoresArea.setEditable(false);
16 userScoresArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
17 userScoresArea.setBackground(new Color(60, 60, 60));
18 userScoresArea.setForeground(Color.WHITE);
19 JScrollPane userScoresScrollPane = new JScrollPane(userScoresArea);
20 userScoresScrollPane.setBorder(BorderFactory.createEmptyBorder());
21 userScoresScrollPane.setViewport().setBackground(new Color(60, 60, 60));
22 userScoresPanel.add(userScoresScrollPane, BorderLayout.CENTER);
23 centerPanel.add(userScoresPanel);
24
25 JPanel leaderboardPanel = new JPanel(new BorderLayout());
26 leaderboardPanel.setBackground(new Color(45, 45, 45));
27 leaderboardPanel.setBorder(BorderFactory.createTitledBorder(
28     BorderFactory.createLineBorder(new Color(75, 75, 75)),
29     "Leaderboards",
30     javax.swing.border.TitledBorder.LEFT,
31     javax.swing.border.TitledBorder.TOP,
32     new Font("Arial", Font.BOLD, 16),
33     Color.WHITE
34));
35 leaderboardArea = new JTextArea();
36 leaderboardArea.setEditable(false);
37 leaderboardArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
38 leaderboardArea.setBackground(new Color(60, 60, 60));
39 leaderboardArea.setForeground(Color.WHITE);
40 JScrollPane leaderboardScrollPane = new JScrollPane(leaderboardArea);
41 leaderboardScrollPane.setBorder(BorderFactory.createEmptyBorder());
42 leaderboardScrollPane.setViewport().setBackground(new Color(60, 60, 60));
43 leaderboardPanel.add(leaderboardScrollPane, BorderLayout.CENTER);
44 centerPanel.add(leaderboardPanel);
45
46 mainPanel.add(centerPanel, BorderLayout.CENTER);
47
```

line 54-99 Panel Tengah (Skor dan Leaderboard)

- Line 54-55: Membuat panel tengah dengan dua kolom: satu untuk skor pengguna dan satu untuk leaderboard.
- Line 57-59: Panel untuk skor pengguna, diberi border berjudul “Your Scores”.
- Line 67-77: Area teks untuk menampilkan skor pengguna dalam bentuk scrollable.
- Line 78-80: Panel untuk leaderboard pengguna terbaik.
- Line 88-99: Menampilkan leaderboard pemain teratas dari dua game.

```
100     JPanel gamePanel = new JPanel(new GridLayout(1, 2, 20, 10));
101     gamePanel.setBackground(new Color(30, 30, 30));
102
103     JButton ticTacToeButton = new JButton();
104     ticTacToeButton.setFont(new Font("Arial", Font.BOLD, 16));
105     ticTacToeButton.setBackground(new Color(255, 152, 0));
106     ticTacToeButton.setForeground(Color.WHITE);
107     ticTacToeButton.setFocusPainted(false);
108     ticTacToeButton.setBorderPainted(false);
109     ticTacToeButton.setPreferredSize(new Dimension(250, 200));
110
111 >     try { ...
112 >     } catch (Exception ex) { ...
113     ticTacToeButton.addActionListener(e -> playTicTacToe());
114     gamePanel.add(ticTacToeButton);
115
116     JButton snakeGameButton = new JButton();
117     snakeGameButton.setFont(new Font("Arial", Font.BOLD, 16));
118     snakeGameButton.setBackground(new Color(76, 175, 80));
119     snakeGameButton.setForeground(Color.WHITE);
120     snakeGameButton.setFocusPainted(false);
121     snakeGameButton.setBorderPainted(false);
122     snakeGameButton.setPreferredSize(new Dimension(250, 200));
123
124 >     try { ...
125 >     } catch (Exception ex) { ...
126     snakeGameButton.addActionListener(e -> playSnakeGame());
127     gamePanel.add(snakeGameButton);
128
129     mainPanel.add(gamePanel, BorderLayout.SOUTH);
130
131     add(mainPanel);
132
133
134
135
136
137
138
139
140 >
141 >
142 >
143 >
144 >
145 >
146 >
147 >
148 >
149 >
150 >
151 >
152 >
153 >
154 >
155 >
156 >
157 >
158 >
159 >
160 >
161 >
162 >
163 >
```

line 99-163 Panel Bawah (Game Buttons)

- Line 101-102: Panel untuk menampung dua tombol: Tic Tac Toe dan Snake Game.
- Line 104-130: Membuat tombol untuk memainkan Tic Tac Toe, mencoba memuat ikon dari folder /assets.

- Line 131-159: Membuat tombol untuk memainkan Snake Game, juga dengan ikon.
- Line 160-163: Menambahkan panel permainan ke bagian bawah jendela dan menampilkan mainPanel.

```

164     loadUserscores();
165     loadLeaderboards();
166 }
```

line 164-167 Load Data dari Database

- Memanggil method untuk mengambil data skor dan leaderboard dari database.

```

167
168     private void loadUserscores() {
169         try (Connection conn = DBConnection.getConnection()) {
170             StringBuilder sb = new StringBuilder();
171
172             String ticTacToeSql = "SELECT COALESCE(SUM(score), 0) as win_count FROM game_scores " +
173                     "WHERE user_id = ? AND game_name = 'Tic Tac Toe'";
174             PreparedStatement ticTacToeStmt = conn.prepareStatement(ticTacToeSql);
175             ticTacToeStmt.setInt(1, user.getUserId());
176             ResultSet ticTacToeRs = ticTacToeStmt.executeQuery();
177
178         if (ticTacToeRs.next()) { ...
179
180             String snakeSql = "SELECT COALESCE(MAX(score), 0) as high_score FROM game_scores " +
181                     "WHERE user_id = ? AND game_name = 'Snake Game'";
182             PreparedStatement snakeStmt = conn.prepareStatement(snakeSql);
183             snakeStmt.setInt(1, user.getUserId());
184             ResultSet snakeRs = snakeStmt.executeQuery();
185
186         if (snakeRs.next()) { ...
187
188             userScoresArea.setText(sb.toString());
189         } catch (SQLException e) {
190             JOptionPane.showMessageDialog(this, "Error loading user scores: " + e.getMessage());
191         }
192     }
193 }
```

line 168-197 Method: loadUserScores()

- Mengambil skor Tic Tac Toe (jumlah menang) dan Snake Game (skor tertinggi) milik user dari database.

```

198     private void loadLeaderboards() {
199         try (Connection conn = DBConnection.getConnection()) {
200             StringBuilder sb = new StringBuilder();
201
202             sb.append("== Tic Tac Toe Top Players ==\n");
203             sb.append(String.format("%-15s %-10s\n", "Player", "Wins"));
204
205             String ticTacToeLeaderboardSql = "SELECT u.nickname, SUM(gs.score) as wins " +
206                 "FROM game_scores gs JOIN users u ON gs.user_id = u.user_id " +
207                 "WHERE gs.game_name = 'Tic Tac Toe' " +
208                 "GROUP BY u.user_id, u.nickname " +
209                 "ORDER BY wins DESC LIMIT 5";
210             PreparedStatement ticTacToeLeaderboardStmt = conn.prepareStatement(ticTacToeLeaderboardSql);
211             ResultSet ticTacToeLeaderboardRs = ticTacToeLeaderboardStmt.executeQuery();
212
213         >         while (ticTacToeLeaderboardRs.next()) { ...
214
215             sb.append("\n== Snake Game Top Players ==\n");
216             sb.append(String.format("%-15s %-10s\n", "Player", "High score"));
217
218             String snakeLeaderboardSql = "SELECT u.nickname, MAX(gs.score) as high_score " +
219                 "FROM game_scores gs JOIN users u ON gs.user_id = u.user_id " +
220                 "WHERE gs.game_name = 'Snake Game' " +
221                 "GROUP BY u.user_id, u.nickname " +
222                 "ORDER BY high_score DESC LIMIT 5";
223             PreparedStatement snakeLeaderboardStmt = conn.prepareStatement(snakeLeaderboardSql);
224             ResultSet snakeLeaderboardRs = snakeLeaderboardStmt.executeQuery();
225
226         >         while (snakeLeaderboardRs.next()) { ...
227
228             leaderboardArea.setText(sb.toString());
229         } catch (SQLException e) { ...
230     }
231
232 }
```

line 198-241 Method: loadLeaderboards()

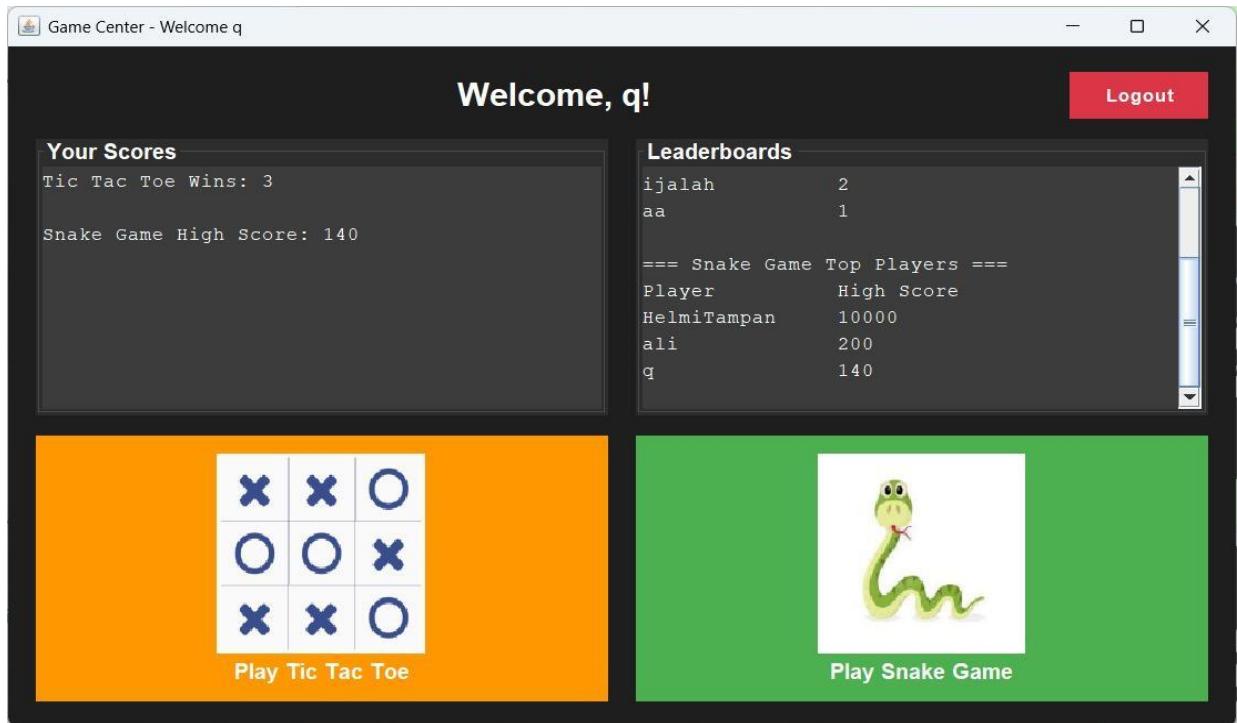
- Mengambil 5 pemain teratas untuk kedua game dari tabel game_scores.

```

241
242     private void playTicTacToe() {
243         new Player2LoginFrame(user).setVisible(true);
244         dispose();
245     }
246
247     private void playSnakeGame() {
248         new GameUlar(user).setVisible(true);
249         dispose();
250     }
251 }
```

line 242-251 Method: playTicTacToe() & playSnakeGame()

- Membuka frame login pemain kedua sebelum main Tic Tac Toe.
- Membuka Snake Game langsung.



Output

- Output menampilkan **dashboard utama**. Tampilan ini muncul setelah pengguna berhasil melakukan login dan berfungsi sebagai pusat navigasi utama bagi pengguna. Tujuan utamanya adalah untuk memberikan informasi yang dipersonalisasi kepada pengguna dan menyediakan akses untuk memulai permainan.

AdminFrame.java

penjelasan code

```
1 package code;
2
3 import code.model.Admin; // Import kelas Admin
4 import code.model.User; // Tetap import User karena User adalah superclas
5 import javax.swing.*;
6 import javax.swing.table.DefaultTableModel;
7
8 import java.awt.*;
9 import java.awt.event.ActionEvent;
10 import java.sql.*;
```

line 1-11 Deklarasi Package dan Import

- code.model.Admin dan code.model.User: mengimpor model dari user dan admin (mewakili entitas user di database).
- javax.swing.*: pustaka GUI seperti JFrame, JButton, JTable, dll.
- java.awt.*: untuk manajemen layout dan komponen grafis.
- java.sql.*: untuk koneksi dan manipulasi database SQL (MySQL, SQLite, dsb).
- DefaultTableModel: digunakan untuk mengelola data tabel di JTable.
- Admin dan User adalah model dari paket code.model.
- line 1-2 Mendeklarasikan bahwa kelas ini berada dalam package code.
- line 3-4 Mengimpor class Admin dan User dari sub-package code.model.
- line 5-11 Mengimpor semua komponen GUI dari Swing, layout dan event handling dari AWT, serta library untuk koneksi dan manipulasi database java.sql.

```

11
12 public class AdminFrame extends JFrame {
13     private JTable usersTable;
14     private JTable scoresTable;
15

```

line 12-15 Deklarasi Kelas dan Konstruktor

- line 12 Mendefinisikan kelas AdminFrame yang merupakan turunan dari JFrame (window utama GUI).
- line 13-14 Dua tabel digunakan untuk menampilkan data users dan scores.
- usersTable: menampilkan data user dari database users.
- scoresTable: menampilkan data skor dari database game_scores.

```

16     // Mengubah parameter konstruktor dari User menjadi Admin
17     public AdminFrame(Admin adminUser) {
18         setTitle("Admin Dashboard - Log-in sebagai " + adminUser.getNickname());
19         setSize(900, 600);
20         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
21         setLocationRelativeTo(null);
22

```

line 16-22

- line 17 Konstruktor yang menerima parameter objek Admin (user dengan hak akses admin).
- line 18-22 Menyiapkan frame dengan ukuran 900x600, judul sesuai nama admin, dan posisi di tengah layar.

```

23 JPanel mainPanel = new JPanel(new BorderLayout());
24 mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
25
26 JPanel topPanel = new JPanel(new BorderLayout());
27 JLabel titleLabel = new JLabel("Admin Dashboard - Log-in sebagai " + adminUser.getNickname(), JLabel.CENTER);
28 titleLabel.setFont(new Font("Arial", Font.BOLD, 18));
29 topPanel.add(titleLabel, BorderLayout.CENTER);
30
31 JButton logoutButton = new JButton("Logout");
32 logoutButton.addActionListener(e -> [
33     new LoginFrame().setVisible(true);
34     dispose();
35 ]);
36 JPanel logoutButtonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
37 logoutButtonPanel.add(logoutButton);
38 topPanel.add(logoutButtonPanel, BorderLayout.EAST);
39
40 mainPanel.add(topPanel, BorderLayout.NORTH);
41

```

line 23-41

- line 23 Panel utama dengan BorderLayout (layout terbagi menjadi Utara, Selatan, Tengah, dll).
- line 24-30 Label judul dengan nama admin dan font besar di tengah atas frame.
- line 31-35 Tombol logout, jika ditekan akan membuka LoginFrame dan menutup AdminFrame.
- line 36-41 Meletakkan tombol logout di kanan atas panel.

```

41 JTabbedPane tabbedPane = new JTabbedPane();
42
43 JPanel usersPanel = new JPanel(new BorderLayout());
44 usersPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
45
46 JButton refreshUsersBtn = new JButton("Refresh");
47 refreshUsersBtn.addActionListener(e -> loadUserData());
48
49 JButton deleteUserBtn = new JButton("Hapus user yang dipilih");
50 deleteUserBtn.addActionListener(this::deleteSelectedUser);
51

```

line 42-51

- Tombol Refresh → Memuat ulang data dari database.
- Tombol Hapus user → Menghapus user yang dipilih dan skornya.
- linw 42-43 Komponen tab yang memungkinkan pemisahan tampilan berdasarkan tab.
- line 44-49 Membuat tombol refresh untuk memuat ulang data user.
- line 50-51 Tombol untuk menghapus user yang dipilih pada tabel.

```
52
53 JPanel usersButtonPanel = new JPanel();
54 usersButtonPanel.add(refreshUsersBtn);
55 usersButtonPanel.add(deleteUserBtn);
56
57 usersPanel.add(usersButtonPanel, BorderLayout.NORTH);
58
59 usersTable = new JTable();
60 usersPanel.add(new JScrollPane(usersTable), BorderLayout.CENTER);
61
62 tabbedPane.addTab("Users", usersPanel);
63
```

line 53-62

- Membuat dan menambahkan tabel user ke panel serta menampilkannya dalam tab "Users".

```
63
64 JPanel scoresPanel = new JPanel(new BorderLayout());
65 scoresPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
66
67 JButton refreshScoresBtn = new JButton("Refresh");
68 refreshScoresBtn.addActionListener(e -> loadScoresData());
69
70 JButton editScoreBtn = new JButton("Edit Score");
71 editScoreBtn.addActionListener(this::editSelectedScore);
72
73 JButton deleteScoreBtn = new JButton("Delete Score");
74 deleteScoreBtn.addActionListener(this::deleteSelectedScore);
75
76 JPanel scoresButtonPanel = new JPanel();
77 scoresButtonPanel.add(refreshScoresBtn);
78 scoresButtonPanel.add(editScoreBtn);
79 scoresButtonPanel.add(deleteScoreBtn);
80
81 scoresPanel.add(scoresButtonPanel, BorderLayout.NORTH);
82
83 scoresTable = new JTable();
84 scoresPanel.add(new JScrollPane(scoresTable), BorderLayout.CENTER);
85
86 tabbedPane.addTab("Scores", scoresPanel);
87
```

line 63-81

- line 64-65 Panel untuk data skor game.
- line 66-69 Tombol refresh untuk memuat data skor terbaru dari database.
- line 70-72 Tombol untuk mengedit skor yang dipilih.
- line 73-75 Tombol untuk menghapus skor dari user.
- line 76-86 Menampilkan tabel skor dalam tab "Scores".

```

87         mainPanel.add(tabbedPane, BorderLayout.CENTER);
88         add(mainPanel);
89
90
91         loadUsersData();
92         loadScoresData();
93     }
94

```

line 88-94

- Menambahkan tab ke frame utama.
- Saat jendela dibuka, data pengguna dan skor akan langsung dimuat dari database.

```

95     private void loadUsersData() {
96         try (Connection conn = DBConnection.getConnection()) {
97             String sql = "SELECT user_id, username, nickname, is_admin FROM users";
98             Statement stmt = conn.createStatement(
99                 ResultSet.TYPE_SCROLL_INSENSITIVE,
100                ResultSet.CONCUR_READ_ONLY
101            );
102            ResultSet rs = stmt.executeQuery(sql);
103
104            usersTable.setModel(DbUtils.resultSetToTableModel(rs));
105        } catch (SQLException e) {
106            JOptionPane.showMessageDialog(this, "Error loading users: " + e.getMessage());
107            e.printStackTrace();
108            usersTable.setModel(new DefaultTableModel());
109        }
110    }
111
112    private void loadScoresData() {
113        try (Connection conn = DBConnection.getConnection()) {
114            String sql = "SELECT gs.score_id, u.username, u.nickname, gs.game_name, gs.score, gs.last_played " +
115                        "FROM game_scores gs JOIN users u ON gs.user_id = u.user_id";
116            Statement stmt = conn.createStatement(
117                ResultSet.TYPE_SCROLL_INSENSITIVE,
118                ResultSet.CONCUR_READ_ONLY
119            );
120            ResultSet rs = stmt.executeQuery(sql);
121
122            scoresTable.setModel(DbUtils.resultSetToTableModel(rs));
123        } catch (SQLException e) {
124            JOptionPane.showMessageDialog(this, "Error loading scores: " + e.getMessage());
125            e.printStackTrace();
126            scoresTable.setModel(new DefaultTableModel());
127        }
128    }
129

```

line 95-129

- line 95-111 Mengambil seluruh data user dari database dan menampilkannya di tabel.
- line 112-129 Mengambil data skor dari tabel game_scores dan menggabungkannya dengan tabel users menggunakan JOIN.

```

130     private void deleteSelectedUser(ActionEvent e) {
131         int selectedRow = usersTable.getSelectedRow();
132     if (selectedRow == -1) { ...
133
134         int userId = (Integer) usersTable.getValueAt(selectedRow, 0);
135         String username = (String) usersTable.getValueAt(selectedRow, 1);
136
137         int confirm = JOptionPane.showConfirmDialog(
138             this,
139             "Apakah yakin menghapus '" + username + "' ?",
140             "Confirm Delete",
141             JOptionPane.YES_NO_OPTION
142         );
143
144         if (confirm == JOptionPane.YES_OPTION) {
145             try (Connection conn = DBConnection.getConnection()) { ...
146                 } catch (SQLException ex) {
147                     JOptionPane.showMessageDialog(this, "Error deleting user: " + ex.getMessage());
148                 }
149             }
150         }
151     }

```

line 130-165 Method deleteSelectedUser()

- Menghapus user yang dipilih dari database, termasuk semua skor terkait, dan memperbarui tampilan.

```

172     private void editSelectedScore(ActionEvent e) {
173         int selectedRow = scoresTable.getSelectedRow();
174         if (selectedRow == -1) {
175             JOptionPane.showMessageDialog(this, "Pilih score yang ingin anda edit");
176             return;
177         }
178
179         int gameId = (Integer) scoresTable.getValueAt(selectedRow, 0);
180         String gameName = (String) scoresTable.getValueAt(selectedRow, 1);
181         int currentScore = (Integer) scoresTable.getValueAt(selectedRow, 2);
182
183         String newScoreStr = JOptionPane.showInputDialog(
184             this,
185             "Edit score for " + gameName + ":\nCurrent score: " + currentScore,
186             currentScore
187         );
188
189         if (newScoreStr != null && !newScoreStr.isEmpty()) {
190             try { ...
191                 } catch (NumberFormatException ex) { ...
192             }
193         }
194     }

```

line 172-212 Method editSelectedScore()

- Memungkinkan admin mengubah skor permainan tertentu.
- Skor baru diinput melalui JOptionPane.
- Mengedit skor game yang dipilih. Jika input valid, data skor di-update ke database.

```

213     private void deleteSelectedScore(ActionEvent e) {
214         int selectedRow = scoresTable.getSelectedRow();
215         if (selectedRow == -1) {
216             JOptionPane.showMessageDialog(this, "Pilih Score yang ingin di hapus");
217             return;
218         }
219
220         int scoreId = (Integer) scoresTable.getValueAt(selectedRow, 0);
221         String gameName = (String) scoresTable.getValueAt(selectedRow, 3);
222         int score = (Integer) scoresTable.getValueAt(selectedRow, 4);
223
224         int confirm = JOptionPane.showConfirmDialog(
225             this,
226             "Hapus score untuk " + gameName + ": " + score + "?",
227             "Confirm Delete",
228             JOptionPane.YES_NO_OPTION
229         );
230
231         if (confirm == JOptionPane.YES_OPTION) {
232             try (Connection conn = DBConnection.getConnection()) {
233                 } catch (SQLException ex) {
234                     JOptionPane.showMessageDialog(this, "Error deleting score: " + ex.getMessage());
235                 }
236             }
237         }
238     }

```

line 213-247 Method deleteSelectedScore()

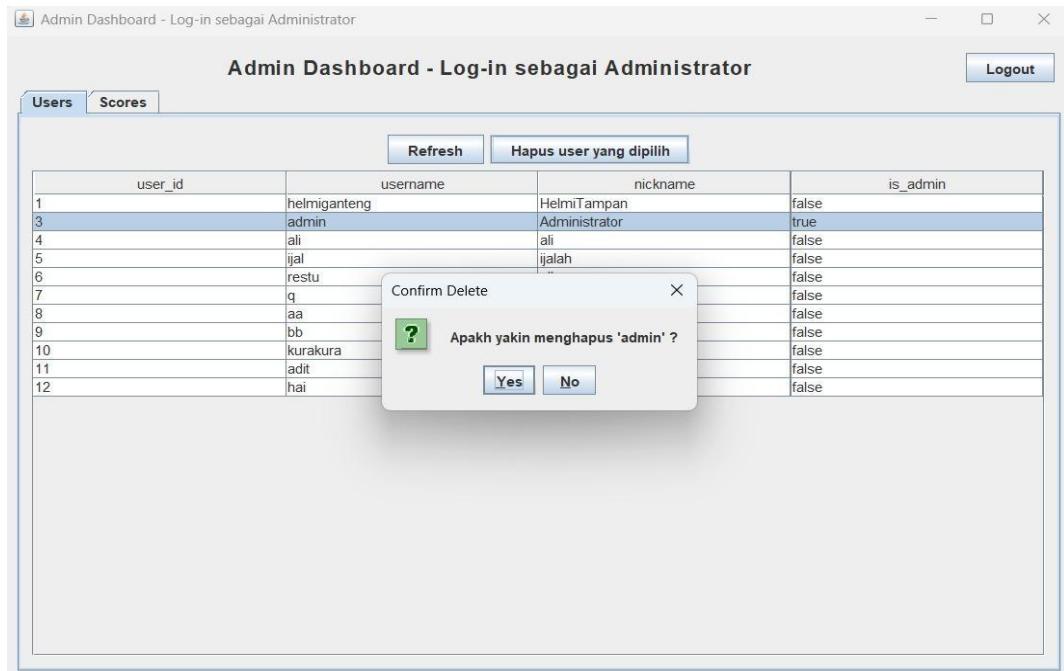
- Menghapus skor berdasarkan score_id dari database dan admin memperbarui tampilan.

Dashboard admin

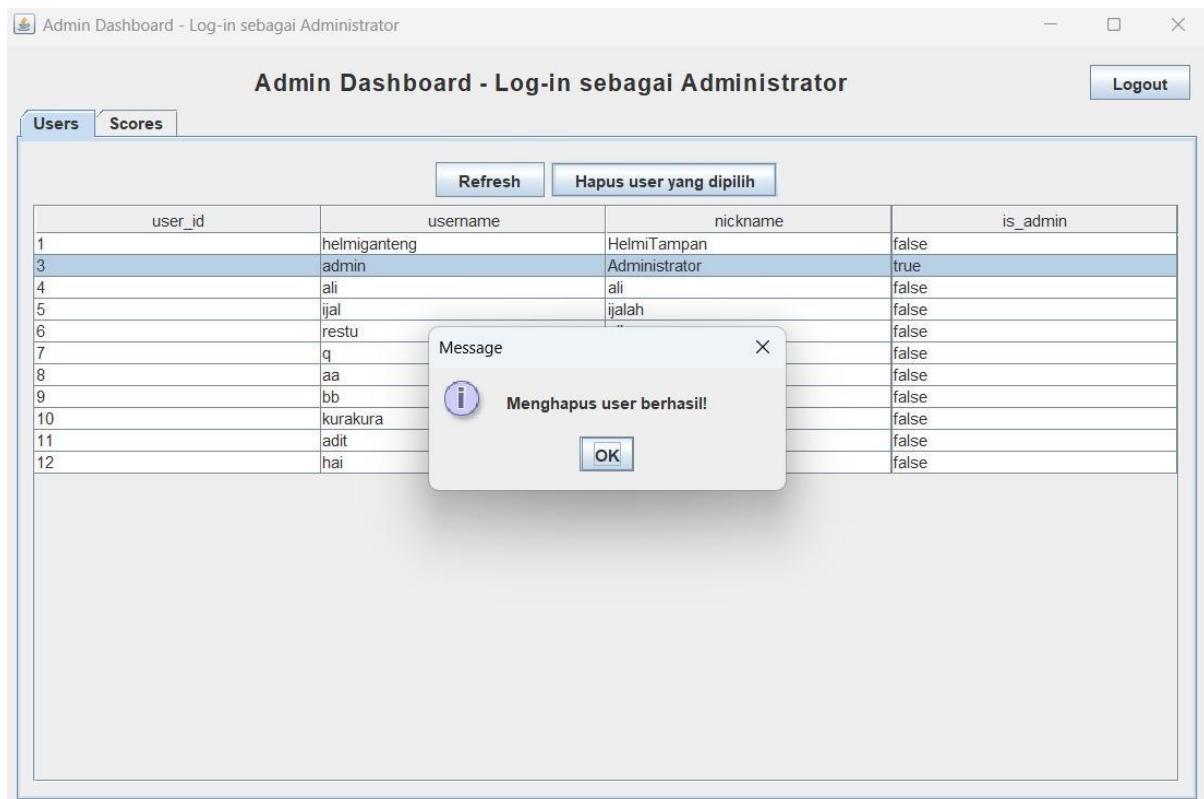
The screenshot shows a Java Swing application window titled "Admin Dashboard - Log-in sebagai Administrator". The window has a title bar with standard window controls (minimize, maximize, close). Below the title bar is a menu bar with "Logout" on the right. The main content area contains two tabs: "Users" (which is selected) and "Scores". Under the "Users" tab, there are two buttons: "Refresh" and "Hapus user yang dipilih". A table displays 12 rows of user data:

user_id	username	nickname	is_admin
1	helmiganteng	HelmiTampan	false
3	admin	Administrator	true
4	ali	ali	false
5	ijal	ijalah	false
6	restu	albar	false
7	q	q	false
8	aa	aa	false
9	bb	bb	false
10	kurakura	kurakura	false
11	adit	ali	false
12	hai	hai	false

pop up jika admin ingin menghapus player



Pop up jika tekan ok saat menghapus player.



Tampilan untuk melihat riwayat permainan user

Admin Dashboard - Log-in sebagai Administrator

Admin Dashboard - Log-in sebagai Administrator

Logout

[Users](#) [Scores](#)

score_id	username	nickname	game_name	score	last_played
6	helmiganteng	HelmiTampan	Snake Game	100	2025-06-03 18:19:34.0
7	helmiganteng	HelmiTampan	Snake Game	10	2025-06-03 18:19:46.0
9	helmiganteng	HelmiTampan	Tic Tac Toe	1	2025-06-03 18:20:07.0
11	helmiganteng	HelmiTampan	Tic Tac Toe	1	2025-06-03 21:52:13.0
12	helmiganteng	HelmiTampan	Snake Game	10000	2025-06-03 22:26:39.0
13	helmiganteng	HelmiTampan	Snake Game	0	2025-06-03 22:26:57.0
27	helmiganteng	HelmiTampan	Tic Tac Toe	1	2025-06-04 00:42:50.0
34	helmiganteng	HelmiTampan	Tic Tac Toe	1	2025-06-04 01:05:31.0
53	helmiganteng	HelmiTampan	Tic Tac Toe	1	2025-06-09 04:28:21.0
54	helmiganteng	HelmiTampan	Snake Game	0	2025-06-09 04:28:29.0
14	ali	ali	Tic Tac Toe	1	2025-06-04 00:04:48.0
15	ali	ali	Snake Game	40	2025-06-04 00:06:09.0
17	ali	ali	Gomoku (15x15)	51	2025-06-04 00:17:51.0
18	ali	ali	Gomoku (15x15)	1	2025-06-04 00:18:17.0
19	ali	ali	Gomoku (15x15)	1	2025-06-04 00:19:57.0
20	ali	ali	Snake Game	0	2025-06-04 00:26:51.0
21	ali	ali	Snake Game	0	2025-06-04 00:26:58.0
22	ali	ali	Snake Game	0	2025-06-04 00:27:06.0
23	ali	ali	Snake Game	140	2025-06-04 00:27:50.0
24	ali	ali	Gomoku (15x15)	1	2025-06-04 00:29:09.0
25	ali	ali	Gomoku (15x15)	1	2025-06-04 00:39:08.0
26	ali	ali	Gomoku (15x15)	1	2025-06-04 00:40:33.0
28	ali	ali	Tic Tac Toe	1	2025-06-04 00:43:14.0
29	ali	ali	Tic Tac Toe 15x15	1	2025-06-04 00:46:40.0
30	ali	ali	Tic Tac Toe 15x15	1	2025-06-04 00:47:06.0

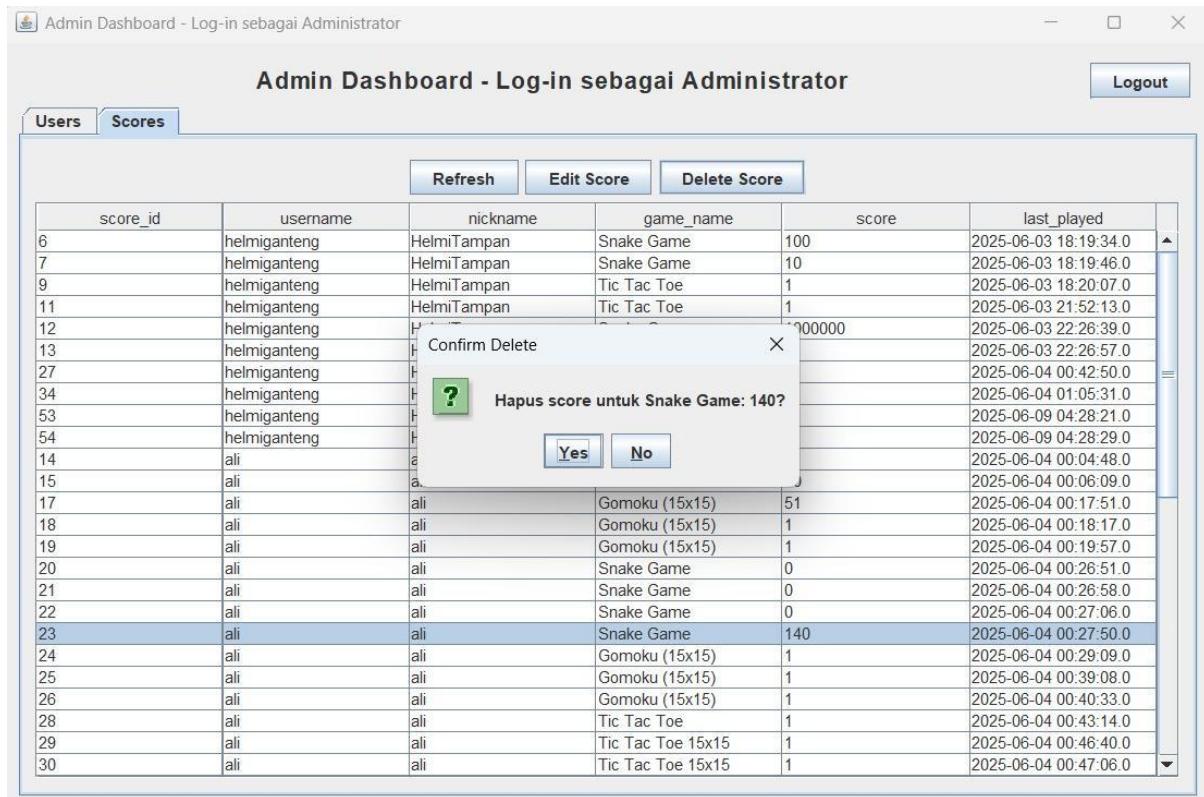
Tampilan jika ingin mengedit score

The screenshot shows the Admin Dashboard interface. At the top, there is a navigation bar with tabs for 'Users' and 'Scores'. Below the navigation bar is a table displaying user scores. A modal dialog box is open over the table, centered on a row where the 'score_id' is 12. The modal has a title 'Edit score for Snake Game:' and a sub-instruction 'Current score: 10000'. It contains a text input field with the value '10000'. At the bottom of the modal are two buttons: 'OK' and 'Cancel'. The background table shows various game records, such as 'Snake Game' and 'Gomoku (15x15)', with scores ranging from 0 to 140 and dates from June 3, 2025, to June 9, 2025.

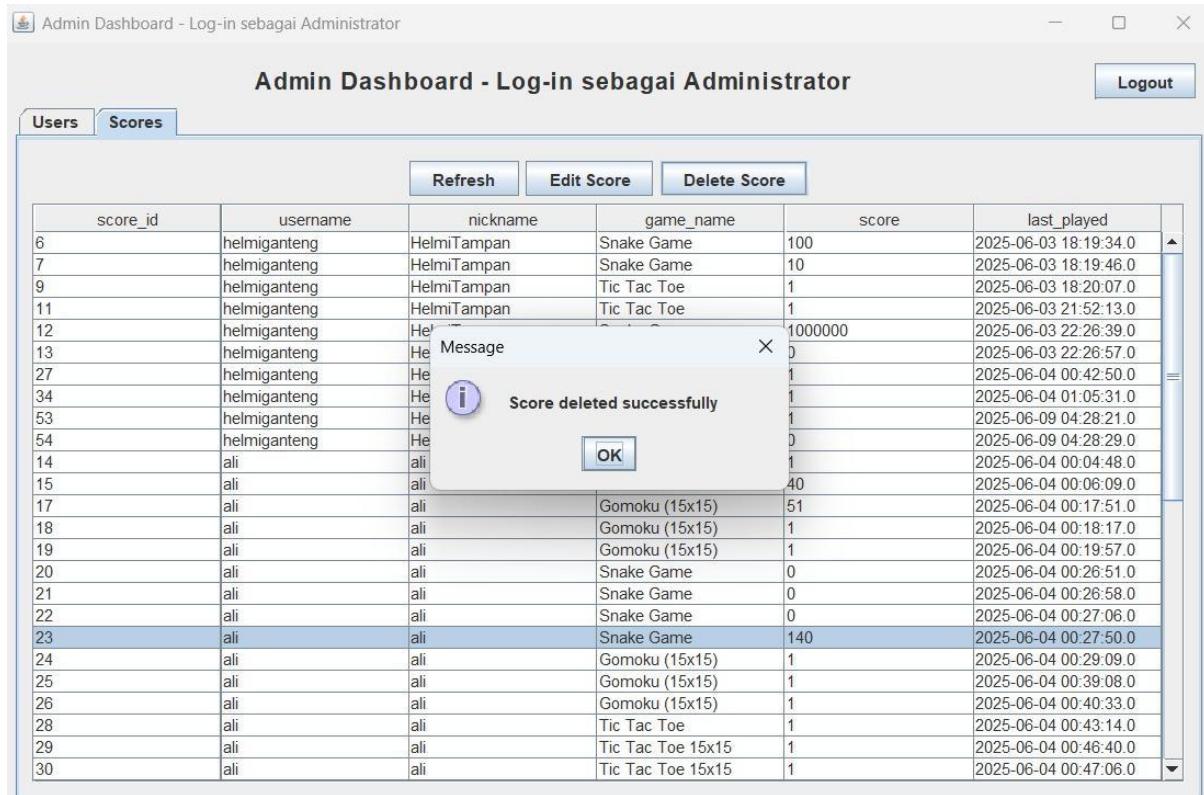
Tampilan popup jika klik oke saat edit score :

The screenshot shows the Admin Dashboard interface, similar to the previous one but with a different modal dialog. The modal has a title 'Message' and a sub-instruction 'Score berhasil di update!'. It contains a single button labeled 'OK'. The background table is the same as in the previous screenshot, showing user scores for various games like Snake Game and Gomoku (15x15). The overall layout is consistent with the first screenshot, showing the 'Admin Dashboard - Log-in sebagai Administrator' title and 'Logout' button at the top.

Pop up hapus score:



pop up saat milih oke saat delete score :



GameTicTacToe.java

```
1  package code;
2
3  import code.model.User;
4  import javax.swing.*;
5  import java.awt.*;
6  import java.awt.event.ActionEvent;
7  import java.sql.*;
8
9  public class GameTicTacToe extends JFrame {
10     private final User player1;
11     private final User player2;
12     private JButton[][] buttons = new JButton[15][15];
13     private boolean xTurn = true;
14     private JLabel statusLabel;
15
16     public GameTicTacToe(User player1, User player2) {
17         this.player1 = player1;
18         this.player2 = player2;
19
20         setTitle(String.format("Tic Tac Toe - %s vs %s", player1.getNickname(), player2.getNickname()));
21         setSize(800, 850);
22         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
23         setLocationRelativeTo(null);
24
25         BackgroundPanel panel = new BackgroundPanel("/assets/board.jpg");
26
27         statusLabel = new JLabel("Giliran " + player1.getNickname() + " (X)", JLabel.CENTER);
28         statusLabel.setFont(new Font("Arial", Font.BOLD, 20));
29         statusLabel.setOpaque(false);
30         statusLabel.setForeground(Color.WHITE);
31         panel.add(statusLabel, BorderLayout.NORTH);
32
33         JPanel gamePanel = new JPanel(new GridLayout(15, 15));
34         gamePanel.setOpaque(false); // transparan agar background terlihat
35         for (int i = 0; i < 15; i++) {
36             for (int j = 0; j < 15; j++) {
37                 buttons[i][j] = new JButton();
38                 buttons[i][j].setFont(new Font("Arial", Font.BOLD, 20));
39                 buttons[i][j].setOpaque(false);
40                 buttons[i][j].setContentAreaFilled(false);
41                 buttons[i][j].setForeground(Color.BLACK);
42                 buttons[i][j].addActionListener(this::buttonClicked);
43                 gamePanel.add(buttons[i][j]);
44             }
45         }
46         panel.add(gamePanel, BorderLayout.CENTER);
47
48         JButton backButton = new JButton("Kembali ke Dashboard");
49         backButton.addActionListener(e -> {
50             new DashboardFrame(player1).setVisible(true);
51             dispose();
52         });
53         panel.add(backButton, BorderLayout.SOUTH);
54
55         add(panel);
56     }
}
```

Baris 1

- package code;
- Deklarasi package bernama code.

Baris 3-7

- import code.modal.User;
- import javax.swing.*;
- import java.awt.*;
- import java.awt.event.ActionEvent;
- import java.awt.event.ActionListener;

Mengimpor class-class dari Java Swing dan AWT untuk membuat tampilan antarmuka (GUI) dan menangani aksi tombol, serta class User dari package lain.

Baris 9-14

- Deklarasi class GameTicToe sebagai turunan dari JFrame, yang akan menjadi jendela utama permainan.
 - private final User player1;
 - private final User player2;
 - private final JButton[][] button = new JButton[3][3];
 - private boolean xTurn = true;
 - private JLabel statusLabel;
- Menyimpan data pemain, tombol-tombol papan 3x3, giliran pemain (X atau O), dan label status.

Baris 21–49

Constructor GameTicToe(User player1, User player2)

- Membuat tampilan dan logika awal permainan.
- Menyimpan nama pemain.
- Mengatur ukuran frame, aksi close, dan layout.
- Membuat panel background gambar papan.
- Membuat label status (giliran siapa).
- Menginisialisasi tombol-tombol papan.
- Menambahkan tombol ke panel.
- Membuat tombol "Kembali ke Dashboard".

- Menambahkan semuanya ke frame.

```

58     private void buttonClicked(ActionEvent e) {
59         JButton button = (JButton) e.getSource();
60
61         if (!button.getText().isEmpty()) return;
62
63         if (xTurn) {
64             button.setText("X");
65             statusLabel.setText("Giliran " + player2.getNickname() + "(O)");
66         } else {
67             button.setText("O");
68             statusLabel.setText("Giliran " + player1.getNickname() + "(X)");
69         }
70
71         xTurn = !xTurn;
72
73         checkWinner();
74     }
75
76     private void checkWinner() {
77         String currentSymbol = xTurn ? "O" : "X";
78
79         for (int row = 0; row < 15; row++) {
80             for (int col = 0; col < 15; col++) {
81                 if (buttons[row][col].getText().equals(currentSymbol)) {
82                     if (checkDirection(row, col, 0, 1, currentSymbol) ||
83                         checkDirection(row, col, 1, 0, currentSymbol) ||
84                         checkDirection(row, col, 1, 1, currentSymbol) ||
85                         checkDirection(row, col, 1, -1, currentSymbol)) {
86
87                         User winningPlayer = currentSymbol.equals("X") ? player1 : player2;
88                         JOptionPane.showMessageDialog(this, winningPlayer.getNickname() + " wins!");
89                         saveScore(winningPlayer);
90                         resetGame();
91                         return;
92                     }
93                 }
94             }
95         }
96
97         if (isBoardFull()) {
98             JOptionPane.showMessageDialog(this, "DRAW!");
99             resetGame();
100        }
101    }
102

```

Baris 51–71

- Method: buttonClicked(ActionEvent)
- Logika saat sebuah tombol papan diklik.
- Jika tombol sudah diisi, keluar dari method.
- Memasukkan simbol “X” atau “O”.

- Mengubah giliran pemain.
- Memanggil method checkWinner() untuk mengecek kemenangan.

Baris 73–93

- Method: checkWinner()
- Cek apakah ada pemenang atau permainan seri.
- Memeriksa semua baris, kolom, dan dua diagonal.
- Jika ada pemenang, simpan skor ke database.
- Jika papan penuh dan tidak ada pemenang, tampilkan pesan “DRAW”.

```

103     private boolean checkDirection(int row, int col, int dx, int dy, String symbol) {
104         int count = 0;
105         for (int i = 0; i < 5; i++) {
106             int r = row + i * dx;
107             int c = col + i * dy;
108             if (r >= 0 && r < 15 && c >= 0 && c < 15 && buttons[r][c].getText().equals(symbol)) {
109                 count++;
110             } else {
111                 break;
112             }
113         }
114         return count == 5;
115     }
116
117     private boolean isBoardFull() {
118         for (int i = 0; i < 15; i++) {
119             for (int j = 0; j < 15; j++) {
120                 if (buttons[i][j].getText().isEmpty()) {
121                     return false;
122                 }
123             }
124         }
125         return true;
126     }

```

Baris 95–110

- Method: checkDirection(int row, int col, int dx, int dy, String symbol)
- Mengecek 3 arah searah dalam 1 garis untuk menemukan simbol yang sama.
- Loop maju dan mundur sejauh 2 langkah.
- Jika menemukan 3 simbol sama → return true.

Baris 112–118

- Method: isBoardFull()
- Cek apakah papan penuh (semua tombol sudah diklik).
- Jika masih ada tombol kosong → return false.
- Jika semua terisi → return true.

Baris 120–125

- Method: resetGame()
- Mengatur ulang papan untuk ronde baru.
- Menghapus semua simbol di tombol.
- Mengatur giliran kembali ke “X”.
- Menampilkan giliran pemain di label status.

```

128     private void resetGame() {
129         for (int i = 0; i < 15; i++) {
130             for (int j = 0; j < 15; j++) {
131                 buttons[i][j].setText("");
132             }
133         }
134         xTurn = true;
135         statusLabel.setText("GILIRAN (X) " + player1.getNickname());
136     }
137
138     private void saveScore(User winner) {
139         try (Connection conn = DBConnection.getConnection()) {
140             String sql = "INSERT INTO game_scores (user_id, game_name, score) VALUES (?, 'Tic Tac Toe', 1) " +
141                         "ON DUPLICATE KEY UPDATE score = score + 1, last_played = CURRENT_TIMESTAMP";
142             PreparedStatement stmt = conn.prepareStatement(sql);
143             stmt.setInt(1, winner.getUserId());
144             stmt.executeUpdate();
145         } catch (SQLException e) {
146             JOptionPane.showMessageDialog(this, "Gagal menyimpan score: " + e.getMessage());
147         }
148     }
149
150     // Background panel class
151     static class BackgroundPanel extends JPanel {
152         private Image backgroundImage;
153
154         public BackgroundPanel(String imagePath) {
155             try {
156                 backgroundImage = new ImageIcon(getClass().getResource(imagePath)).getImage();
157             } catch (Exception e) {
158                 JOptionPane.showMessageDialog(this, "gagal menggunakan background image: " + e.getMessage());
159             }
160             setLayout(new BorderLayout());
161         }
162
163         @Override
164         protected void paintComponent(Graphics g) {
165             super.paintComponent(g);
166             if (backgroundImage != null) {
167                 g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
168             }
169         }
170     }
171 }

```

Baris 127–137

- Method: saveScore(User winner)
- Menyimpan data pemenang ke database MySQL.
- Menyisipkan nama game, skor, dan nama pemenang ke tabel game_scores.
- Jika gagal, tampilkan pesan error menggunakan JOptionPane.

Baris 139–167

- Class: BackgroundPanel
- Class khusus untuk membuat panel dengan background gambar.
- Membaca gambar dari resource (board.jpg).
- Override method paintComponent() untuk menggambar background ke panel.



Output:

Output menampilkan tampilan permainan Tic Tac Toe versi besar (15×15 grid) dengan latar belakang papan kayu. Di bagian atas tertulis “Giliran q (X)” yang menandakan giliran pemain dengan simbol “X” (dalam hal ini pemain bernama “q”). Terdapat juga tombol “Kembali ke Dashboard” di bawah untuk keluar dari permainan.



Output:

Sebuah pop-up dialog “q wins!” muncul sebagai notifikasi bahwa pemain bernama “q” memenangkan permainan.

GameUlar

```
1 package code;
2
3 import code.model.User;
4 import javax.swing.*;
5 import java.awt.*;
6 import java.awt.event.*;
7 import java.sql.*;
8 import java.util.ArrayList;
9 import java.util.Random;
```

import bertugas "membawa masuk" (mengakses) kelas-kelas yang diperlukan dari paket-paket lain di Java ke dalam kelas GameUlar ini, sehingga Anda bisa menggunakannya tanpa perlu menulis nama paket lengkapnya setiap kali. Ini termasuk:

- code.model.User: Untuk menggunakan objek User yang kemungkinan berisi data pengguna.

- javax.swing.*: Untuk membangun antarmuka grafis (GUI) permainan, menyediakan komponen seperti jendela, tombol, label, dan panel.
- java.awt.*: Untuk fungsionalitas grafis dasar dan objek geometris seperti titik (Point), warna (Color), dan gambar (Image).
- java.awt.event.*: Untuk menangani interaksi pengguna, khususnya input keyboard (penekanan tombol panah) yang mengontrol ular.
- java.sql.*: Untuk berinteraksi dengan basis data (melalui JDBC) guna menyimpan dan mengambil skor permainan.
- java.util.ArrayList dan java.util.Random: Untuk struktur data seperti daftar (digunakan untuk tubuh ular) dan generasi angka acak (digunakan untuk penempatan makanan).

```

11 public class GameUlar extends JFrame {
12     private final User user;
13     private static final int TILE_SIZE = 25;
14     private static final int WIDTH = 20;
15     private static final int HEIGHT = 20;
16     private static final int GAME_SPEED = 150;
17
18     private ArrayList<Point> snake;
19     private Point food;
20     private char direction = 'R';
21     private boolean isRunning = false;
22     private Timer timer;
23     private int score = 0;
24     private JLabel scoreLabel;
25     private Image backgroundImage;
26

```

- Public class GameUlar extends JFrame menandakan bahwa class GameUlar akan menyimpan seluruh fitur untuk game ular ini
- private final User user; Variabel ini menyimpan objek User yang mewakili pemain saat ini. Karena ini final, nilai user akan ditetapkan saat objek GameUlar dibuat dan tidak bisa diubah setelahnya. Ini penting untuk mengaitkan permainan dan skor dengan pemain yang benar.
- private static final int TILE_SIZE = 25; Ini adalah konstanta yang menentukan ukuran satu "ubin" atau kotak di papan permainan, yaitu 25 piksel. Karena static dan final, nilai ini sama untuk semua instance GameUlar dan tidak akan berubah.
- private static final int WIDTH = 20; Ini adalah konstanta yang menentukan lebar papan permainan dalam jumlah ubin, yaitu 20 ubin.
- private static final int HEIGHT = 20; Ini adalah konstanta yang menentukan tinggi papan permainan dalam jumlah ubin, yaitu 20 ubin.

- private static final int GAME_SPEED = 150; Ini adalah konstanta yang menentukan kecepatan permainan. Angka ini (150) adalah durasi dalam milidetik antara setiap pergerakan ular. Semakin kecil angkanya, semakin cepat ular bergerak.
- private ArrayList<Point> snake; Ini adalah daftar (ArrayList) yang akan menyimpan koordinat (Point) dari setiap segmen tubuh ular. Point adalah kelas yang menyimpan pasangan koordinat x dan y. Elemen pertama dalam daftar ini (snake.get(0)) akan selalu menjadi kepala ular.
- private Point food; Variabel ini akan menyimpan koordinat (Point) dari posisi makanan di papan permainan
- private char direction = 'R'; // U, D, L, R Variabel ini menyimpan arah pergerakan ular saat ini. Karakter 'R' berarti Kanan. Komentar // U, D, L, R menunjukkan nilai: 'U' (Atas), 'D' (Bawah), 'L' (Kiri).
- private boolean isRunning = false; Ini adalah bendera (flag) boolean yang menunjukkan apakah permainan sedang berjalan (true) atau sudah berakhir/belum dimulai (false).
- private Timer timer; Ini adalah objek Timer dari Swing yang akan digunakan untuk mengatur laju permainan. Timer akan secara berkala memicu tindakan (pergerakan ular, dll.) sesuai dengan GAME_SPEED.
- private int score = 0; Variabel ini menyimpan skor pemain saat ini dalam permainan.
- private JLabel scoreLabel; Ini adalah objek JLabel dari Swing yang digunakan untuk menampilkan skor pemain di antarmuka grafis permainan.
- private Image backgroundImage; Variabel ini akan menyimpan objek Image yang digunakan sebagai latar belakang papan permainan. Jika gambar latar belakang berhasil dimuat, ia akan ditampilkan di belakang ubin permainan.

```

27     public GameUlar(User user) {
28         this.user = user;
29         setTitle("Snake Game - " + user.getNickname());
30         setSize(WIDTH * TILE_SIZE + 10, HEIGHT * TILE_SIZE + 70);
31         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
32         setLocationRelativeTo(null);
33
34         try {
35             backgroundImage = new ImageIcon(getClass().getResource(name:"/assets/snakeboard.png")).getImage();
36         } catch (Exception e) {
37             System.err.println("Error loading background image for Snake Game: " + e.getMessage());
38             backgroundImage = null;
39         }
40
41         JPanel panel = new JPanel(new BorderLayout());
42
43         scoreLabel = new JLabel(text:"Score: 0", JLabel.CENTER);
44         scoreLabel.setFont(new Font(name:"Arial", Font.BOLD, size:16));
45         scoreLabel.setForeground(Color.WHITE);
46         panel.add(scoreLabel, BorderLayout.NORTH);
47
48         GamePanel gamePanel = new GamePanel();
49         panel.add(gamePanel, BorderLayout.CENTER);
50
51         JButton backButton = new JButton(text:"Back to Dashboard");
52         backButton.addActionListener(e -> {
53             new DashboardFrame(user).setVisible(true);
54             dispose();
55         });
56         panel.add(backButton, BorderLayout.SOUTH);
57
58         add(panel);
59
60         startGame();
61

```

Line 27-32: Konstruktor GameUlar:

- Menginisialisasi game dengan objek User, mengatur judul jendela (Snake Game - [nama pengguna]), ukuran jendela, perilaku saat ditutup, dan memposisikan jendela di tengah layar.

Line 34-39: Pemuatan Gambar Latar Belakang:

- Mencoba memuat snakeboard.png sebagai gambar latar belakang. Jika gagal, akan mencetak pesan error ke konsol dan mengatur backgroundImage menjadi null.

Line 41-45: Pengaturan Panel Utama & Skor:

- Membuat JPanel utama dengan BorderLayout. Menambahkan JLabel (scoreLabel) dengan teks "Score: 0", font Arial Bold 16pt, warna putih, dan menempatkannya di bagian atas (NORTH) panel.

Line 48-49: Pengaturan Panel Game:

- Membuat GamePanel (tempat logika dan gambar game ular berada) dan menempatkannya di bagian tengah (CENTER) panel utama.

Line 51-57: Pengaturan Tombol "Back to Dashboard":

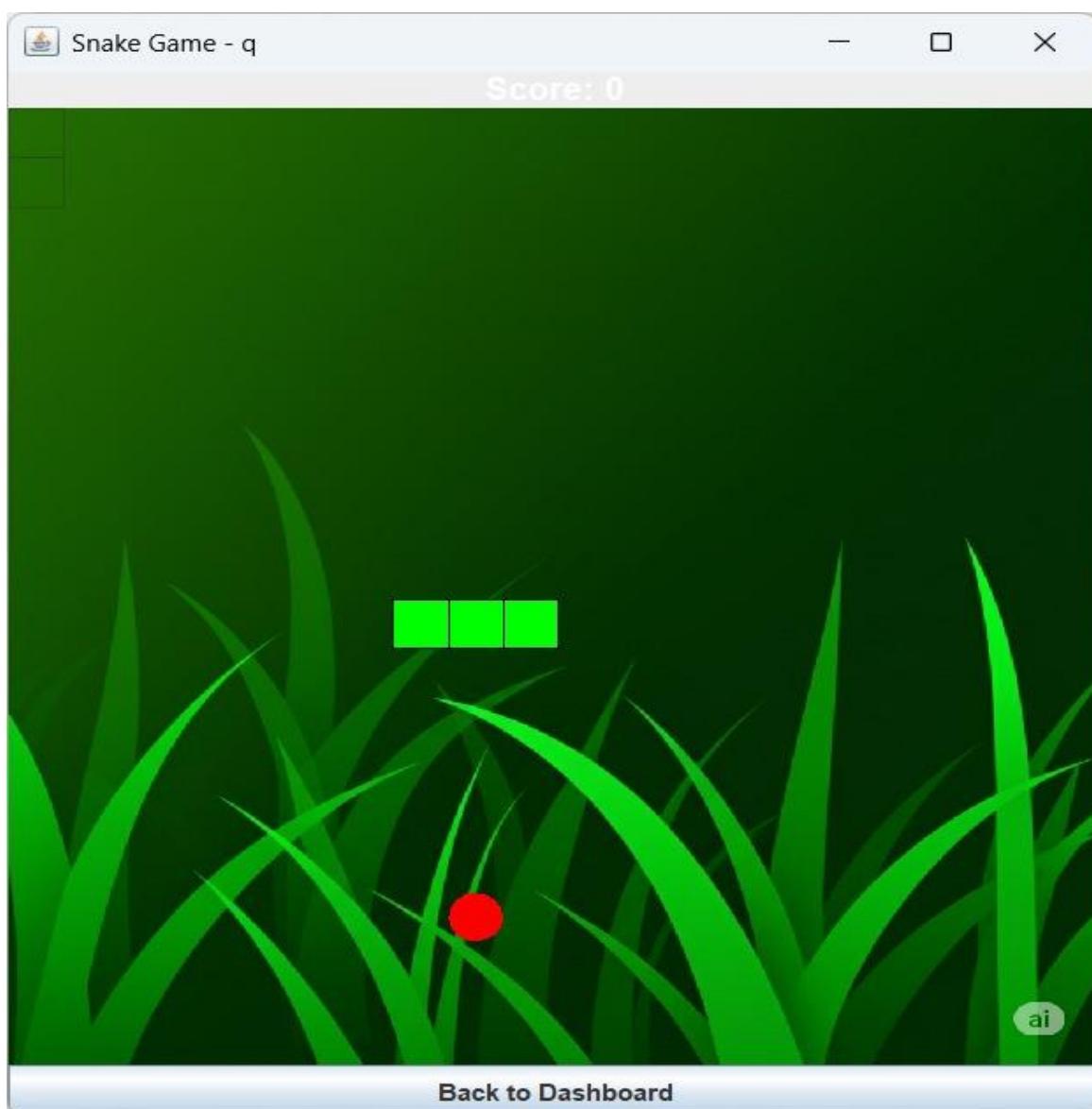
- Membuat JButton dengan teks "Back to Dashboard". Menambahkan *listener* yang saat diklik akan menampilkan DashboardFrame baru untuk User yang sama dan menutup jendela GameUlar saat ini. Tombol ini ditempatkan di bagian bawah (SOUTH) panel utama.

Line 59: Penambahan Panel ke Frame:

- Menambahkan seluruh panel utama (yang berisi skor, game panel, dan tombol) ke dalam jendela GameUlar.

Line 61: Mulai Game:

- Memanggil metode startGame() untuk memulai logika permainan ular.



Output : Menghasilkan sebuah board permainan game ular yang sudah di atur sedeminikan rupa.

```
53     private void startGame() {
54         snake = new ArrayList<Point>();
55         snake.add(new Point(x:5, y:5));
56         snake.add(new Point(x:4, y:5));
57         snake.add(new Point(x:3, y:5));
58
59         spawnFood();
60         direction = 'R';
61         score = 0;
62         scoreLabel.setText(text:"Score: 0");
63         isRunning = true;
64
65         timer = new Timer(GAME_SPEED, e -> gameLoop());
66         timer.start();
67
68         addKeyListener(new KeyAdapter() {
69             @Override
70             public void keyPressed(KeyEvent e) {
71                 switch (e.getKeyCode()) {
72                     case KeyEvent.VK_UP:
73                         if (direction != 'D') direction = 'U';
74                         break;
75                     case KeyEvent.VK_DOWN:
76                         if (direction != 'U') direction = 'D';
77                         break;
78                     case KeyEvent.VK_LEFT:
79                         if (direction != 'R') direction = 'L';
80                         break;
81                     case KeyEvent.VK_RIGHT:
82                         if (direction != 'L') direction = 'R';
83                         break;
84                 }
85             }
86         });
87
88         setFocusable(focusable:true);
89         requestFocusInWindow();
90     }
91 }
```

Line 63: Deklarasi Metode startGame():

- Mendefinisikan metode privat bernama startGame() yang tidak mengembalikan nilai. Metode ini akan menginisialisasi dan memulai semua aspek penting dari permainan ular.

Line 64: Inisialisasi Ular:

- Membuat ArrayList baru yang akan menyimpan koordinat (objek Point) dari setiap segmen tubuh ular.

Line 65-67: Posisi Awal Ular:

- Menambahkan tiga Point ke ArrayList snake, menetapkan posisi awal ular dengan kepala di (5,5) dan dua segmen tubuh lainnya di (4,5) dan (3,5).

Line 69: Memunculkan Makanan:

- Memanggil metode spawnFood() (tidak terlihat di sini) untuk menempatkan makanan di lokasi acak pada papan permainan.

Line 70: Arah Awal:

- Mengatur arah pergerakan awal ular ke kanan ('R').

Line 71:

- Inisialisasi Skor: Mengatur skor awal permainan menjadi 0.

Line 72: Perbarui Label Skor:

- Memperbarui teks pada scoreLabel di antarmuka pengguna untuk menampilkan "Score: 0".

Line 73: Status Permainan Berjalan:

- Mengatur variabel boolean isRunning menjadi true, yang menandakan bahwa permainan sudah aktif.

Line 75: Inisialisasi Timer Game:

- Membuat objek Timer baru. Timer ini akan memicu eksekusi metode gameLoop() (yang kemungkinan berisi logika pembaruan permainan) secara berkala setiap GAME_SPEED milidetik.

Line 76: Mulai Timer:

- Mengaktifkan timer yang baru dibuat, memulai siklus permainan yang berulang.

Line 78-96: Penanganan Input Keyboard:

- Menambahkan KeyListener ke komponen untuk mendengarkan penekanan tombol keyboard.

Line 80: Metode keyPressed:

- Ini adalah metode yang akan dipanggil setiap kali tombol keyboard ditekan.

Line 81-94: Logika Perubahan Arah:

- Menggunakan pernyataan switch untuk memeriksa tombol panah yang ditekan (VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT). Arah ular (direction) akan diubah sesuai tombol yang ditekan, tetapi hanya jika perubahan arah tersebut tidak berarti ular langsung berbalik 180 derajat (misalnya, tidak bisa langsung dari 'U' ke 'D').

Line 98: Fokus Komponen:

- Mengatur komponen ini agar dapat menerima fokus input, yang esensial agar KeyListener dapat mendeteksi penekanan tombol.

Line 99: Meminta Fokus Input:

- Meminta fokus input ke komponen ini, memastikan bahwa ia akan menjadi yang pertama menerima input keyboard ketika jendela aplikasi aktif.

```
1 private void gameLoop() {
2     if (!isRunning) return;
3
4     // Move snake
5     Point head = snake.get(0);
6     Point newHead;
7
8     switch (direction) {
9         case 'U':
10            newHead = new Point(head.x, head.y - 1);
11            break;
12        case 'D':
13            newHead = new Point(head.x, head.y + 1);
14            break;
15        case 'L':
16            newHead = new Point(head.x - 1, head.y);
17            break;
18        case 'R':
19            newHead = new Point(head.x + 1, head.y);
20            break;
21        default:
22            return;
23    }
24
25    // Check collision with walls
26    if (newHead.x < 0 || newHead.x >= WIDTH || newHead.y < 0 || newHead.y >= HEIGHT) {
27        gameOver();
28        return;
29    }
30
31    for (Point segment : snake) {
32        if (segment.equals(newHead)) {
33            gameOver();
34            return;
35        }
36    }
37
38    snake.add(0, newHead);
39
40
41    if (newHead.equals(food)) {
42        score += 10;
43        scoreLabel.setText("Score: " + score);
44        spawnFood();
45    } else {
46        snake.remove(snake.size() - 1);
47    }
48
49    repaint();
50}
51 }
```

Line 102: Deklarasi Metode gameLoop():

- Mendefinisikan metode privat bernama gameLoop(). Metode ini adalah inti dari logika permainan ular, yang akan dijalankan secara berkala oleh Timer untuk memperbarui status permainan.

Line 103: Pemeriksaan Status Game:

- Memeriksa apakah permainan sedang berjalan (!isRunning). Jika isRunning adalah false (artinya game telah berakhir atau dihentikan), metode akan segera berhenti (return), mencegah pembaruan game lebih lanjut.

Line 106: Mengambil Kepala Ular:

- Mendapatkan Point yang merepresentasikan posisi kepala ular saat ini (elemen pertama dalam ArrayList snake).

Line 107-124: Menentukan Posisi Kepala Baru:

- Blok switch ini menghitung posisi newHead (kepala ular yang akan datang) berdasarkan direction (arah pergerakan ular saat ini).
 - Line 110: Jika arah 'U' (Up), newHead bergerak satu langkah ke atas ($y - 1$).
 - Line 113: Jika arah 'D' (Down), newHead bergerak satu langkah ke bawah ($y + 1$).
 - Line 116: Jika arah 'L' (Left), newHead bergerak satu langkah ke kiri ($x - 1$).
 - Line 119: Jika arah 'R' (Right), newHead bergerak satu langkah ke kanan ($x + 1$).
 - Line 122-123: default return: Jika arah tidak valid (kasus ini seharusnya tidak terjadi dalam permainan normal), metode akan berhenti.

Line 127-130: Pemeriksaan Tabrakan Dinding:

- Mengecek apakah newHead (posisi kepala ular yang baru) berada di luar batas papan permainan (kurang dari 0 atau melebihi WIDTH/HEIGHT). Jika terjadi tabrakan, gameOver() dipanggil dan metode berhenti.

Line 133-137: Pemeriksaan Tabrakan Diri Sendiri:

- Melakukan iterasi melalui setiap segment tubuh ular. Jika newHead bertabrakan dengan salah satu segment tubuh ular (ular menggigit dirinya sendiri), gameOver() dipanggil dan metode berhenti.

Line 139: Menambahkan Kepala Baru:

- Menambahkan newHead ke posisi pertama (indeks 0) dari ArrayList snake. Ini secara efektif menggerakkan kepala ular ke depan.

Line 142-146: Logika Makan Makanan:

- Mengecek apakah newHead berada di posisi yang sama dengan food.
 - Line 142: Jika ular memakan makanan (newHead.equals(food)):
 - Line 143: Skor ditambahkan 10.
 - Line 144: scoreLabel diperbarui untuk menampilkan skor baru.
 - Line 145: spawnFood() dipanggil lagi untuk menempatkan makanan baru.
 - Line 146: Jika ular tidak memakan makanan (else):
 - Line 147: Segmen terakhir dari ular ($\text{snake.size}() - 1$) dihapus. Ini memberikan ilusi pergerakan ular tanpa pertumbuhan.

Line 150: Menggambar Ulang:

- Memanggil repaint(). Ini memberi tahu sistem Swing untuk menggambar ulang komponen (termasuk ular dan makanan) di layar, merefleksikan posisi terbaru mereka.

Line 151: Penutup metode gameLoop().

```
154     private void spawnFood() {  
155         Random random = new Random();  
156         int x, y;  
157  
158         do {  
159             x = random.nextInt(WIDTH);  
160             y = random.nextInt(HEIGHT);  
161             food = new Point(x, y);  
162         } while (snake.contains(food));  
163     }  
164 }
```

Line 154: Deklarasi Metode spawnFood():

- Mendefinisikan metode privat bernama spawnFood() yang tidak mengembalikan nilai. Metode ini bertanggung jawab untuk menghasilkan lokasi baru untuk makanan ular.

Line 155: Inisialisasi Objek Random:

- Membuat objek Random baru. Objek ini akan digunakan untuk menghasilkan angka acak yang diperlukan untuk menentukan koordinat makanan.

Line 156: Deklarasi Variabel Koordinat:

- Mendeklarasikan dua variabel integer, x dan y, yang akan menyimpan koordinat horizontal dan vertikal makanan.

Line 158: Blok do-while Loop:

- Memulai sebuah perulangan do-while. Loop ini akan terus dijalankan selama kondisi while bernilai true. Ini memastikan bahwa makanan tidak muncul di tempat yang sudah ditempati ular.
 - Line 159: Koordinat X Acak: Menghasilkan angka acak untuk koordinat x antara 0 (inklusif) dan WIDTH (eksklusif). WIDTH kemungkinan adalah lebar papan permainan.
 - Line 160: Koordinat Y Acak: Menghasilkan angka acak untuk koordinat y antara 0 (inklusif) dan HEIGHT (eksklusif). HEIGHT kemungkinan adalah tinggi papan permainan.
 - Line 161: Membuat Objek Makanan: Membuat objek Point baru dengan koordinat x dan y yang dihasilkan secara acak, dan menetapkannya ke variabel food.

Line 162: Kondisi while:

- Kondisi untuk perulangan do-while. Perulangan akan terus berjalan (true) selama snake (list segmen tubuh ular) sudah mengandung (.contains()) food yang baru saja dihasilkan. Ini memastikan makanan tidak muncul di atas ular.

Line 163: Penutup Metode spawnFood():

- Menutup blok metode spawnFood().

```
154     private void spawnFood() {
155         Random random = new Random();
156         int x, y;
157
158         do {
159             x = random.nextInt(WIDTH);
160             y = random.nextInt(HEIGHT);
161             food = new Point(x, y);
162         } while (snake.contains(food));
163     }
164
165     private void gameOver() {
166         isRunning = false;
167         timer.stop();
168         JOptionPane.showMessageDialog(this, "Game Over! Your score: " + score);
169         saveScore();
170     }
171 }
```

Line 165: Deklarasi Metode gameOver():

- Mendefinisikan metode privat bernama gameOver() yang tidak mengembalikan nilai. Metode ini akan dipanggil ketika permainan berakhir (misalnya, ular menabrak dinding atau dirinya sendiri).

Line 166: Menghentikan Permainan:

- Mengatur variabel boolean isRunning menjadi false. Ini akan menghentikan eksekusi gameLoop() di siklus berikutnya, secara efektif menghentikan pergerakan ular dan logika permainan.

Line 167: Menghentikan Timer:

- Memanggil stop() pada objek timer. Ini akan menghentikan Timer agar tidak lagi memicu gameLoop() secara berkala.

Line 168: Menampilkan Pesan Game Over:

- Menampilkan kotak dialog pesan menggunakan JOptionPane.showMessageDialog(). Pesan yang ditampilkan adalah "Game Over! Your score: " diikuti dengan skor akhir pemain. this menunjukkan bahwa dialog akan berpusat pada komponen tempat gameOver() dipanggil.

Line 169: Menyimpan Skor:

- Memanggil metode saveScore(). Metode ini (yang tidak terlihat di potongan kode ini) kemungkinan bertanggung jawab untuk menyimpan skor pemain ke database atau file.

Line 170: Penutup Metode gameOver(): Menutup blok metode gameOver().



Output : Menghasilkan popup message yang berisi Game Over! dan skor yang user dapatkan setelah bermain Game Ular.

```
172     private void saveScore() {
173         try (Connection conn = DBConnection.getConnection()) {
174             String sql = "INSERT INTO game_scores (user_id, game_name, score) VALUES (?, 'Snake Game', ?) " +
175                         "ON DUPLICATE KEY UPDATE score = GREATEST(score, VALUES(score)), last_played = CURRENT_TIMESTAMP";
176
177             PreparedStatement stmt = conn.prepareStatement(sql);
178             stmt.setInt(parameterIndex:1, user.getUserId());
179             stmt.setInt(parameterIndex:2, score);
180             stmt.executeUpdate();
181         } catch (SQLException e) {
182             JOptionPane.showMessageDialog(this, "Error saving score: " + e.getMessage());
183         }
184     }
```

Line 172: Deklarasi Metode saveScore():

- Mendefinisikan metode privat bernama saveScore() yang tidak mengembalikan nilai. Metode ini bertanggung jawab untuk menyimpan skor permainan ke dalam database.

Line 173: Blok try

- : Memulai blok try, yang digunakan untuk menampung kode yang mungkin menimbulkan SQLException (kesalahan terkait database).
 - Line 173: Mendapatkan Koneksi Database: Memanggil DBConnection.getConnection() untuk mendapatkan objek Connection ke database. DBConnection kemungkinan adalah kelas utilitas yang mengelola koneksi database.

Line 174-175: Query SQL:

- Mendefinisikan String sql yang berisi perintah SQL untuk menyisipkan (INSERT) skor ke dalam tabel game_scores.

- INSERT INTO game_scores (user_id, game_name, score) VALUES (?, 'Snake Game', ?): Ini adalah bagian pertama dari query yang mencoba menyisipkan user_id, nama game ('Snake Game'), dan score. Tanda tanya (?) adalah *placeholder* untuk nilai yang akan disisipkan nanti.
- ON DUPLICATE KEY UPDATE score = GREATEST(score, VALUES(score)), last_played = CURRENT_TIMESTAMP": Ini adalah bagian ON DUPLICATE KEY UPDATE yang digunakan jika ada duplikat user_id dan game_name (misalnya, jika pengguna sudah memiliki skor untuk game ini).
- score = GREATEST(score, VALUES(score)): Akan memperbarui kolom score menjadi nilai yang lebih besar antara skor yang sudah ada dan skor baru yang akan disisipkan. Ini memastikan bahwa hanya skor tertinggi yang disimpan.
- last_played = CURRENT_TIMESTAMP: Akan memperbarui kolom last_played dengan tanggal dan waktu saat ini.

Line 177: Mempersiapkan Pernyataan:

- Membuat objek PreparedStatement dari objek Connection dan query sql. PreparedStatement lebih aman dari serangan *SQL injection* dan lebih efisien untuk eksekusi berulang.

Line 178: Mengatur Parameter User ID:

- Mengatur nilai integer untuk *placeholder* pertama (?) dalam query SQL dengan user.getUserId(). Ini mendapatkan ID pengguna dari objek user yang terkait dengan game.

Line 179: Mengatur Parameter Skor:

- Mengatur nilai integer untuk *placeholder* kedua (?) dalam query SQL dengan nilai score saat ini.

Line 180: Mengeksekusi Pembaruan:

- Mengeksekusi perintah SQL. executeUpdate() digunakan untuk perintah SQL yang memodifikasi data (INSERT, UPDATE, DELETE).

Line 181: Blok catch:

- Menangkap SQLException jika ada kesalahan yang terjadi selama operasi database di blok try.
 - Line 182: Menampilkan Pesan Error: Menampilkan kotak dialog pesan menggunakan JOptionPane.showMessageDialog() yang berisi pesan "Error saving score: " diikuti dengan detail pesan error dari objek SQLException.

Line 183: Penutup Blok try-catch.

Line 184: Penutup Metode saveScore().

```
186 private class GamePanel extends JPanel {  
187     @Override  
188     protected void paintComponent(Graphics g) {  
189         super.paintComponent(g);  
190  
191         if (backgroundImage != null) {  
192             g.drawImage(backgroundImage, x:0, y:0, getWidth(), getHeight(), this);  
193         } else {  
194             g.setColor(Color.BLACK);  
195             g.fillRect(x:0, y:0, WIDTH * TILE_SIZE, HEIGHT * TILE_SIZE);  
196         }  
197         // Draw snake  
198         for (Point p : snake) {  
199             g.setColor(Color.GREEN);  
200             g.fillRect(p.x * TILE_SIZE, p.y * TILE_SIZE, TILE_SIZE, TILE_SIZE);  
201             g.setColor(Color.BLACK); // Border snake  
202             g.drawRect(p.x * TILE_SIZE, p.y * TILE_SIZE, TILE_SIZE, TILE_SIZE);  
203         }  
204         // Draw food  
205         if (food != null) {  
206             g.setColor(Color.RED);  
207             g.fillOval(food.x * TILE_SIZE, food.y * TILE_SIZE, TILE_SIZE, TILE_SIZE);  
208         }  
209         g.setColor(new Color(r:50, g:50, b:50, a:100));  
210         for (int i = 0; i < WIDTH; i++) {  
211             for (int j = 0; j < HEIGHT; j++) {  
212                 g.drawRect(i * TILE_SIZE, j * TILE_SIZE, TILE_SIZE, TILE_SIZE);  
213             }  
214         }  
215     }  
216 }  
217 }  
218 }
```

Line 186: Deklarasi Kelas GamePanel:

- Mendefinisikan kelas GamePanel yang merupakan subkelas dari JPanel. Ini berarti GamePanel adalah komponen UI yang dapat digambar, dan akan digunakan untuk menampilkan visual permainan.

Line 187: Deklarasi Metode paintComponent():

- Ini adalah metode yang di-override dari kelas JPanel. Metode ini adalah tempat di mana semua operasi penggambaran kustom untuk GamePanel dilakukan. Parameter Graphics g adalah objek yang digunakan untuk melakukan operasi penggambaran.

Line 189: Memanggil super.paintComponent():

- Penting untuk selalu memanggil metode paintComponent() dari superkelas. Ini membersihkan area gambar dan melakukan tugas-tugas standar lainnya yang

diperlukan sebelum penggambaran kustom Anda.

Line 191-196: Menggambar Latar Belakang:

- Blok if-else ini menentukan apakah akan menggambar gambar latar belakang atau warna solid.
 - Line 191: if (backgroundImage != null): Memeriksa apakah objek backgroundImage ada (tidak null).
 - Line 192: g.drawImage(backgroundImage, x:0, y:0, getWidth(), getHeight(), this);: Jika backgroundImage ada, gambar tersebut digambar mengisi seluruh area GamePanel.
 - Line 193: else {}: Jika backgroundImage adalah null (misalnya, gagal dimuat).
 - Line 195: g.setColor(Color.BLACK);: Mengatur warna gambar menjadi hitam.
 - Line 196: g.fillRect(x:0, y:0, WIDTH * TILE_SIZE, HEIGHT * TILE_SIZE);: Mengisi seluruh area papan permainan dengan warna hitam. WIDTH dan HEIGHT kemungkinan adalah dimensi papan dalam unit "tile", dan TILE_SIZE adalah ukuran setiap "tile" dalam piksel.

Line 198-204: Menggambar Ular:

- Bagian ini bertanggung jawab untuk menggambar setiap segmen ular.
 - Line 198: for (Point p : snake) {}: Melakukan iterasi melalui setiap Point (p) di ArrayList snake (yang merepresentasikan setiap segmen tubuh ular).
 - Line 199: g.setColor(Color.GREEN);: Mengatur warna gambar menjadi hijau untuk mengisi segmen ular.
 - Line 200: g.fillRect(p.x * TILE_SIZE, p.y * TILE_SIZE, TILE_SIZE, TILE_SIZE);: Mengisi persegi panjang hijau pada posisi segmen ular. Koordinat dihitung dengan mengalikan posisi Point (p.x, p.y) dengan TILE_SIZE untuk mengubahnya dari unit grid ke piksel.
 - Line 201: g.setColor(Color.BLACK); // Border snake: Mengatur warna gambar menjadi hitam untuk menggambar batas.
 - Line 202: g.drawRect(p.x * TILE_SIZE, p.y * TILE_SIZE, TILE_SIZE, TILE_SIZE);: Menggambar persegi panjang hitam di sekeliling segmen ular, membuat efek batas.

Line 206-209: Menggambar Makanan:

- Bagian ini menggambar makanan jika ada.
 - Line 206: if (food != null) {}: Memeriksa apakah objek food ada (tidak null).

- Line 207: `g.setColor(Color.RED);`: Mengatur warna gambar menjadi merah untuk makanan.
- Line 208: `g.fillOval(food.x * TILE_SIZE, food.y * TILE_SIZE, TILE_SIZE, TILE_SIZE);`: Mengisi bentuk oval (lingkaran) merah di posisi makanan. Koordinat dihitung serupa dengan ular.

Player2Login.java

```

1 package code;
2
3 import code.model.User;
4 import javax.swing.*;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.sql.*;
8

```

Line 1: Deklarasi Package: `package code;`

- Line ini mendeklarasikan bahwa file Java ini adalah bagian dari package code. Package digunakan untuk mengorganisir kelas-kelas Java dan mencegah konflik penamaan.

Line 3: Import Kelas User: `import code.model.User;`

- Line ini mengimpor kelas User dari package code.model. Ini berarti kelas-kelas dalam file ini dapat menggunakan kelas User tanpa harus menulis nama package lengkapnya (code.model.User) setiap kali. Kelas User kemungkinan mendefinisikan struktur data untuk pengguna aplikasi.

Line 4: Import Semua Kelas Swing: `import javax.swing.*;`

- Line ini mengimpor semua kelas dari package javax.swing. Package Swing menyediakan komponen GUI (Graphical User Interface) untuk aplikasi Java, seperti JFrame, JPanel, JButton, JLabel, JOptionPane, dan Timer.

Line 5: Import Semua Kelas AWT: `import java.awt.*;`

- Line ini mengimpor semua kelas dari package java.awt. AWT (Abstract Window Toolkit) adalah toolkit GUI asli Java. Beberapa kelas AWT masih sering digunakan bersama Swing, seperti Graphics, Color, Font, dan Point.

Line 6: Import Kelas ActionEvent: `import java.awt.event.ActionEvent;`

- Line ini mengimpor kelas ActionEvent secara spesifik dari package java.awt.event. ActionEvent digunakan untuk event yang dihasilkan oleh komponen AWT atau

Swing, seperti ketika tombol diklik. Meskipun `java.awt.*` sudah diimpor di Line 5, mengimpor `ActionEvent` secara eksplisit mungkin dilakukan karena kebiasaan atau kebutuhan spesifik di beberapa IDE, meskipun seringkali tidak wajib jika `java.awt.*` sudah ada.

Line 7: Import Semua Kelas SQL: `import java.sql.*;`

- Line ini mengimpor semua kelas dari package `java.sql`. Package ini menyediakan API Java Database Connectivity (JDBC) untuk berinteraksi dengan database, termasuk kelas seperti `Connection`, `PreparedStatement`, `SQLException`, dll. Ini mengindikasikan bahwa aplikasi ini akan berinteraksi dengan database

```
9  public class Player2LoginFrame extends JFrame {
10    private final User player1;
11    private JTextField usernameField;
12    private JPasswordField passwordField;
13
14    public Player2LoginFrame(User player1) {
15      this.player1 = player1;
16      setTitle(title:"Game Center - Login Player 2");
17      setSize(width:900, height:550);
18      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
19      setLocationRelativeTo(c:null);
20
21      // Mengatur warna latar belakang untuk seluruh frame
22      getContentPane().setBackground(new Color(r:30, g:30, b:30));
23
24      // Menggunakan BorderLayout untuk frame utama: judul di atas, form di tengah
25      setLayout(new BorderLayout());
```

Line 9: Deklarasi Kelas Player2LoginFrame: `public class Player2LoginFrame extends JFrame {`

- Line ini mendeklarasikan kelas publik bernama `Player2LoginFrame`. Kata kunci `extends JFrame` menunjukkan bahwa `Player2LoginFrame` adalah subclass dari `JFrame`. Ini berarti `Player2LoginFrame` akan menjadi jendela aplikasi GUI.

Line 10: Deklarasi Variabel player1: `private final User player1;`

- Line ini mendeklarasikan variabel instance privat dan final bernama `player1` dengan tipe `User`. `final` berarti variabel ini hanya dapat diinisialisasi sekali (melalui konstruktor) dan nilainya tidak dapat diubah setelah itu. Ini kemungkinan menyimpan objek `User` yang merepresentasikan pemain pertama.

Line 11: Deklarasi Variabel usernameField: `private JTextField usernameField;`

- Line ini mendeklarasikan variabel instance privat bernama usernameField dengan tipe JTextField. Ini akan menjadi komponen input teks di GUI tempat pengguna (emain kedua) akan memasukkan nama pengguna mereka.

Line 12: Deklarasi Variabel passwordField: private JPasswordField passwordField;

- Line ini mendeklarasikan variabel instance privat bernama passwordField dengan tipe JPasswordField. Ini adalah komponen input teks khusus untuk kata sandi, yang biasanya menyembunyikan karakter yang diketik.

Line 14: Konstruktor Player2LoginFrame: public Player2LoginFrame(User player1) {

- Ini adalah konstruktor untuk kelas Player2LoginFrame. Ketika objek Player2LoginFrame baru dibuat, konstruktor ini akan dipanggil. Ia menerima satu argumen: sebuah objek User yang disebut player1.

Line 15: Inisialisasi player1: this.player1 = player1;

- Line ini menginisialisasi variabel instance player1 (yang dideklarasikan di Line 10) dengan objek player1 yang diterima sebagai argumen konstruktor.

Line 16: Mengatur Judul Jendela: setTitle(title:"Game Center - Login Player 2");

- Line ini mengatur judul jendela (frame) aplikasi menjadi "Game Center - Login Player 2". Ini akan muncul di bilah judul jendela.

Line 17: Mengatur Ukuran Jendela: setSize(width:900, height:550);

- Line ini mengatur ukuran jendela aplikasi. Lebarnya diatur menjadi 900 piksel dan tingginya menjadi 550 piksel.

Line 18: Mengatur Operasi Tutup Default:

setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

- Line ini mengatur perilaku jendela saat tombol tutup (ikon 'X' di bilah judul) diklik. JFrame.DISPOSE_ON_CLOSE berarti jendela akan ditutup dan sumber dayanya dibebaskan, tetapi aplikasi Java secara keseluruhan mungkin tidak berakhir jika ada jendela lain yang masih terbuka.

Line 19: Memposisikan Jendela di Tengah: setLocationRelativeTo(c:null);

- Line ini memposisikan jendela di tengah layar. Ketika argumennya null, jendela akan berpusat pada layar.

Line 22: Mengatur Warna Latar Belakang Konten Pane:

getContentPane().setBackground(new Color(r:30, g:30, b:30));

- Line ini mendapatkan *content pane* dari JFrame (area tempat komponen GUI akan ditambahkan) dan mengatur warna latar belakangnya menjadi abu-abu sangat gelap (new Color(30, 30, 30)).

Line 25: Mengatur Layout Manager Utama: setLayout(new BorderLayout());

- Line ini mengatur *layout manager* untuk JFrame ini menjadi BorderLayout. BorderLayout membagi ruang menjadi lima area: NORTH (atas), SOUTH (bawah), EAST (kanan), WEST (kiri), dan CENTER (tengah). Komentar di Line 24 mengindikasikan bahwa ini digunakan untuk menempatkan judul di atas dan form di tengah

```

26 // panel player 2 login
27 JPanel headerPanel = new JPanel();
28 headerPanel.setBackground(new Color(r:30, g:30, b:30));
29 headerPanel.setBorder(BorderFactory.createEmptyBorder(top:30, left:0, bottom:20, right:0));
30
31 JLabel title = new JLabel(text:"PLAYER 2 LOGIN");
32 title.setFont(new Font(name:"Arial", Font.BOLD, size:36));
33 title.setForeground(Color.WHITE);
34 headerPanel.add(title);
35 add(headerPanel, BorderLayout.NORTH);
36
37 // --- Panel Form Login (konten utama) ---
38 JPanel formPanel = new JPanel(new GridBagLayout());
39 formPanel.setBackground(new Color(r:45, g:45, b:45));
40 formPanel.setBorder(BorderFactory.createEmptyBorder(top:40, left:80, bottom:40, right:80));
41
42 GridBagConstraints gbc = new GridBagConstraints();
43 gbc.insets = new Insets(top:10, left:0, bottom:10, right:0);
44 gbc.fill = GridBagConstraints.HORIZONTAL;
45 gbc.anchor = GridBagConstraints.CENTER;
46
47 // Label Username
48 JLabel userLabel = new JLabel(text:"USERNAME");
49 userLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:16));
50 userLabel.setForeground(new Color(r:200, g:200, b:200));
51 gbc.gridx = 0;
52 gbc.gridy = 0;
53 gbc.anchor = GridBagConstraints.WEST;
54 formPanel.add(userLabel, gbc);
55

```

Line 28: Inisialisasi headerPanel: JPanel headerPanel = new JPanel();

- Membuat objek JPanel baru bernama headerPanel. Panel ini akan digunakan sebagai wadah untuk judul login pemain 2.

Line 29: Mengatur Latar Belakang headerPanel: headerPanel.setBackground(new Color(r:30, g:30, b:30));

- Mengatur warna latar belakang headerPanel menjadi abu-abu sangat gelap (Color(30, 30, 30)).

Line 30: Mengatur Border headerPanel:

headerPanel.setBorder(BorderFactory.createEmptyBorder(top:30, left:0, bottom:20, right:0));

- Mengatur border kosong (padding) untuk headerPanel. Ini menambahkan ruang kosong di sekitarnya: 30 piksel di atas, 0 di kiri, 20 di bawah, dan 0 di kanan.

Line 32: Inisialisasi title Label: JLabel title = new JLabel(text:"PLAYER 2 LOGIN");

- Membuat objek JLabel baru bernama title dengan teks "PLAYER 2 LOGIN". Ini akan berfungsi sebagai judul utama di bagian atas frame.

Line 33: Mengatur Font title: title.setFont(new Font(name:"Arial", font:Font.BOLD, size:36));

- Mengatur font untuk title label. Fontnya adalah "Arial", gayanya tebal (Font.BOLD), dan ukurannya 36.

Line 34: Mengatur Warna Teks title: title.setForeground(Color.WHITE);

- Mengatur warna teks title menjadi putih.

Line 35: Menambahkan title ke headerPanel: headerPanel.Add(title);

- Menambahkan title label sebagai komponen ke headerPanel.

Line 36: Menambahkan headerPanel ke Frame Utama: add(headerPanel, BorderLayout.NORTH);

- Menambahkan headerPanel (yang berisi judul "PLAYER 2 LOGIN") ke frame utama. Karena frame utama menggunakan BorderLayout, headerPanel ditempatkan di area NORTH (atas).

Line 39: Inisialisasi formPanel: JPanel formPanel = new JPanel(new GridBagLayout());

- Membuat objek JPanel baru bernama formPanel. Panel ini akan menampung elemen-elemen formulir login (username, password, tombol). Pentingnya, ia menggunakan GridBagLayout sebagai *layout manager*-nya, yang memungkinkan penempatan komponen yang fleksibel dalam grid, dengan kontrol yang granular atas posisi dan ukuran.

Line 40: Mengatur Latar Belakang formPanel: formPanel.setBackground(new Color(r:45, g:45, b:45));

- Mengatur warna latar belakang formPanel menjadi abu-abu sedikit lebih terang (Color(45, 45, 45)) dari header.

Line 41: Mengatur Border formPanel:

```
formPanel.setBorder(BorderFactory.createEmptyBorder(top:40, left:80, bottom:40, right:80));
```

- Mengatur border kosong (padding) untuk formPanel. Ini menambahkan ruang kosong 40 piksel di atas dan bawah, serta 80 piksel di kiri dan kanan.

Line 43: Inisialisasi GridBagConstraints: GridBagConstraints gbc = new GridBagConstraints();

- Membuat objek GridBagConstraints baru bernama gbc. Objek ini digunakan dengan GridBagLayout untuk menentukan bagaimana komponen akan ditempatkan dan perlakunya dalam grid.

Line 44: Mengatur Inset gbc: gbc.insets = new Insets(top:10, left:0, bottom:10, right:0);

- Mengatur insets (padding eksternal) untuk komponen yang akan menggunakan gbc. Ini menambahkan ruang kosong 10 piksel di atas dan bawah setiap komponen yang ditambahkan dengan gbc ini, dan 0 di kiri dan kanan.

Line 45: Mengatur fill gbc: gbc.fill = GridBagConstraints.HORIZONTAL;

- Mengatur fill properti gbc menjadi HORIZONTAL. Ini berarti komponen akan mengisi ruang horizontal yang tersedia di sel grid-nya jika ada ruang kosong.

Line 46: Mengatur anchor gbc: gbc.anchor = GridBagConstraints.CENTER;

- Mengatur anchor properti gbc menjadi CENTER. Ini berarti jika komponen tidak mengisi seluruh ruang yang tersedia di sel grid-nya, komponen akan diposisikan di tengah sel tersebut.

Line 49: Inisialisasi userLabel: JLabel userLabel = new JLabel(text:"USERNAME");

- Membuat objek JLabel baru bernama userLabel dengan teks "USERNAME".

Line 50: Mengatur Font userLabel: userLabel.setFont(new Font(name:"Arial", font:Font.PLAIN, size:16));

- Mengatur font untuk userLabel. Fontnya "Arial", gayanya PLAIN (normal), dan ukurannya 16.

Line 51: Mengatur Warna Teks userLabel: userLabel.setForeground(new Color(r:200, g:200, b:200))

- Mengatur warna teks userLabel menjadi abu-abu muda (Color(200, 200, 200)).

Line 52: Mengatur gridx untuk gbc: gbc.gridx = 0;

- Mengatur posisi kolom (gridx) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi kolom pertama (indeks 0).

Line 53: Mengatur gridy untuk gbc: gbc.gridy = 0;

- Mengatur posisi baris (gridy) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi baris pertama (indeks 0).

Line 54: Mengatur anchor untuk gbc: gbc.anchor = GridBagConstraints.WEST;

- Mengatur anchor untuk komponen ini ke WEST (kiri). Ini berarti userLabel akan diposisikan di sisi kiri sel grid-nya.

Line 55: Menambahkan userLabel ke formPanel: formPanel.add(userLabel, gbc);

- Menambahkan userLabel ke formPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

```

57     // Field Username
58     usernameField = new JTextField(columns:25);
59     usernameField.setFont(new Font(name:"Arial", Font.PLAIN, size:18));
60     usernameField.setBackground(Color.WHITE);
61     usernameField.setForeground(Color.BLACK);
62     usernameField.setBorder(BorderFactory.createCompoundBorder(
63         BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1),
64         BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15)
65     ));
66     gbc.gridx = 1;
67     gbc.weightx = 1.0;
68     gbc.anchor = GridBagConstraints.CENTER;
69     formPanel.add(usernameField, gbc);
70
71     // Label Password
72     JLabel passLabel = new JLabel(text:"PASSWORD");
73     passLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:16));
74     passLabel.setForeground(new Color(r:200, g:200, b:200));
75     gbc.gridx = 2;
76     gbc.insets = new Insets(top:20, left:0, bottom:10, right:0);
77     gbc.anchor = GridBagConstraints.WEST;
78     formPanel.add(passLabel, gbc);

```

Line 58: Inisialisasi usernameField: usernameField = new JTextField(columns:25);

- Membuat objek JTextField baru bernama usernameField dengan lebar 25 kolom karakter. Ini adalah tempat pengguna akan memasukkan nama pengguna.

Line 59: Mengatur Font usernameField: usernameField.setFont(new Font(name:"Arial", font:Font.PLAIN, size:18));

- Mengatur font untuk usernameField. Fontnya "Arial", gaya PLAIN (normal), dan ukurannya 18.

Line 60: Mengatur Latar Belakang usernameField:
usernameField.setBackground(Color.WHITE);

- Mengatur warna latar belakang usernameField menjadi putih.

Line 61: Mengatur Warna Teks usernameField:
usernameField.setForeground(Color.BLACK);

- Mengatur warna teks (input pengguna) di usernameField menjadi hitam.

Line 62-65: Mengatur Border usernameField:
usernameField.setBorder(BorderFactory.createCompoundBorder(...));

- Mengatur border gabungan (CompoundBorder) untuk usernameField. Ini menggabungkan dua jenis border:
 - Line 63: BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1): Membuat border garis tipis (1 piksel) berwarna abu-abu muda (Color(200, 200, 200)) di sekeliling field.
 - Line 64: BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15): Membuat border kosong (padding internal) di dalam border garis, menambahkan ruang 10 piksel di atas dan bawah, serta 15 piksel di kiri dan kanan teks. Ini memberikan jarak antara teks dan batas field.

Line 66: Mengatur gridy untuk gbc: gbc.gridx = 1;

- Mengatur posisi baris (gridy) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi baris kedua (indeks 1), di bawah label username.

Line 67: Mengatur weightx untuk gbc: gbc.weightx = 1.0;

- Mengatur weightx untuk gbc menjadi 1.0. Ini berarti komponen ini akan mengambil semua ruang horizontal ekstra yang tersedia di kolomnya. Ini penting untuk membuat field memanjang secara proporsional saat jendela diubah ukurannya.

Line 68: Mengatur anchor untuk gbc: gbc.anchor = GridBagConstraints.CENTER;

- Mengatur anchor properti gbc menjadi CENTER. Ini berarti jika komponen tidak mengisi seluruh ruang yang tersedia di sel grid-nya (setelah fill dan weightx diterapkan), komponen akan diposisikan di tengah sel tersebut.

Line 69: Menambahkan usernameField ke formPanel: formPanel.add(usernameField, gbc);

- Menambahkan usernameField ke formPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

Line 72: Inisialisasi passLabel: JLabel passLabel = new JLabel(text:"PASSWORD");

- Membuat objek JLabel baru bernama passLabel dengan teks "PASSWORD".

Line 73: Mengatur Font passLabel: passLabel.setFont(new Font(name:"Arial", font:Font.PLAIN, size:16));

- Mengatur font untuk passLabel. Fontnya "Arial", gaya PLAIN (normal), dan ukurannya 16.

Line 74: Mengatur Warna Teks passLabel: passLabel.setForeground(new Color(r:200, g:200, b:200));

- Mengatur warna teks passLabel menjadi abu-abu muda (Color(200, 200, 200)).

Line 75: Mengatur gridy untuk gbc: gbc.gridx = 2;

- Mengatur posisi baris (gridy) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi baris ketiga (indeks 2), di bawah username field.

Line 76: Mengatur Inset gbc (ulang): gbc.insets = new Insets(top:20, left:0, bottom:10, right:0);

- Penting: Line ini menimpa (override) pengaturan insets sebelumnya. Sekarang, komponen yang ditambahkan dengan gbc ini akan memiliki padding 20 piksel di atas, 0 di kiri, 10 di bawah, dan 0 di kanan. Ini kemungkinan untuk memberikan jarak vertikal yang lebih besar antara usernameField dan passLabel.

Line 77: Mengatur anchor untuk gbc: gbc.anchor = GridBagConstraints.WEST;

- Mengatur anchor untuk komponen ini ke WEST (kiri). Ini berarti passLabel akan diposisikan di sisi kiri sel grid-nya.

Line 78: Menambahkan passLabel ke formPanel: formPanel.add(passLabel, gbc);

- Menambahkan passLabel ke formPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

```

80 // Field Password
81 passwordField = new JPasswordField(columns:25);
82 passwordField.setFont(new Font(name:"Arial", Font.PLAIN, size:18));
83 passwordField.setBackground(Color.WHITE);
84 passwordField.setForeground(Color.BLACK);
85 passwordField.setCaretColor(Color.BLACK);
86 passwordField.setBorder(BorderFactory.createCompoundBorder(
87     BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1),
88     BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15)
89 ));
90 gbc.gridx = 3;
91 gbc.insets = new Insets(top:0, left:0, bottom:30, right:0);
92 gbc.anchor = GridBagConstraints.CENTER;
93 formPanel.add(passwordField, gbc);

94 // Panel Tombol Login/Cancel
95 JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, hgap:20, vgap:0));
96 buttonPanel.setBackground(new Color(r:45, g:45, b:45));
97
98

```

Line 81: Inisialisasi passwordField: passwordField = new JPasswordField(columns:25);

- Membuat objek JPasswordField baru bernama passwordField dengan lebar 25 kolom karakter. Ini adalah tempat pengguna akan memasukkan kata sandi mereka, dengan karakter yang disembunyikan.

Line 82: Mengatur Font passwordField: passwordField.setFont(new Font(name:"Arial", font:Font.PLAIN, size:18));

- Mengatur font untuk passwordField. Fontnya "Arial", gaya PLAIN (normal), dan ukurannya 18.

Line 83: Mengatur Latar Belakang passwordField:

```
passwordField.setBackground(Color.WHITE);
```

- Mengatur warna latar belakang passwordField menjadi putih.

Line 84: Mengatur Warna Teks passwordField:

```
passwordField.setForeground(Color.BLACK);
```

- Mengatur warna teks (input pengguna) di passwordField menjadi hitam.

Line 85: Mengatur Warna Caret passwordField:

```
passwordField.setCaretColor(Color.BLACK);
```

- Mengatur warna *caret* (kursor yang berkedip) di passwordField menjadi hitam.

Line 86-89: Mengatur Border passwordField:

```
passwordField.setBorder(BorderFactory.createCompoundBorder(...));
```

- Mengatur border gabungan (CompoundBorder) untuk passwordField, sama seperti usernameField. Ini menggabungkan:
 - Line 87: BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1): Border garis tipis (1 piksel) berwarna abu-abu muda di sekeliling field.
 - Line 88: BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15): Padding internal yang menambahkan ruang 10 piksel di atas dan bawah, serta 15 piksel di kiri dan kanan teks di dalam field.

Line 90: Mengatur gridy untuk gbc: gbc.gridy = 3;

- Mengatur posisi baris (gridy) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi baris keempat (indeks 3), di bawah password label.

Line 91: Mengatur Inset gbc (ulang): gbc.insets = new Insets(top:0, left:0, bottom:30, right:0);

- Penting: Line ini menimpa pengaturan insets sebelumnya. Sekarang, komponen yang ditambahkan dengan gbc ini akan memiliki padding 0 piksel di atas dan kiri/kanan, serta 30 piksel di bawah. Ini memberikan jarak vertikal yang lebih besar antara passwordField dan komponen berikutnya (tombol).

Line 92: Mengatur anchor untuk gbc: gbc.anchor = GridBagConstraints.CENTER;

- Mengatur anchor properti gbc menjadi CENTER. Ini berarti komponen akan diposisikan di tengah sel grid-nya.

Line 93: Menambahkan passwordField ke formPanel: formPanel.add(passwordField, gbc);

- Menambahkan passwordField ke formPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

Line96: Inisialisasi buttonPanel: JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, hgap:20, vgap:0));

- Membuat objek JPanel baru bernama buttonPanel. Panel ini akan digunakan untuk menampung tombol-tombol. Ia menggunakan FlowLayout yang menata komponen dalam satu baris. FlowLayout.CENTER menengahkan komponen secara horizontal, hgap:20 memberikan celah horizontal 20 piksel antar komponen, dan vgap:0 berarti tidak ada celah vertikal antar baris (karena hanya ada satu baris).

Line 97: Mengatur Latar Belakang buttonPanel: buttonPanel.setBackground(new Color(r:45, g:45, b:45));

- Mengatur warna latar belakang buttonPanel menjadi abu-abu gelap (Color(45, 45, 45)), sama dengan formPanel.

```

98     // Login button
99     JButton loginButton = new JButton(text:"LOGIN");
100    loginButton.setFont(new Font(name:"Arial", Font.BOLD, size:18));
101    loginButton.setBackground(new Color(r:76, g:175, b:80));
102    loginButton.setForeground(Color.WHITE);
103    loginButton.setFocusPainted(b:false);
104    loginButton.setBorderPainted(b:false);
105    loginButton.setPreferredSize(new Dimension(width:150, height:45));
106    buttonPanel.add(loginButton);
107    loginButton.addActionListener(this::performLogin);
108
109   // Cancel button
110   JButton cancelButton = new JButton(text:"CANCEL");
111   cancelButton.setFont(new Font(name:"Arial", Font.BOLD, size:18));
112   cancelButton.setBackground(new Color(r:220, g:53, b:69));
113   cancelButton.setForeground(Color.WHITE);
114   cancelButton.setFocusPainted(b:false);
115   cancelButton.setBorderPainted(b:false);
116   cancelButton.setPreferredSize(new Dimension(width:150, height:45));
117   buttonPanel.add(cancelButton);
118   cancelButton.addActionListener(e -> {
119     new DashboardFrame(player1).setVisible(b:true);
120     dispose();
121   });
122

```

Line 100-108 : mengatur style untuk login button.

Line 111-117 : mengatur style tombol Cancel button.

Line 119-122: Menambahkan ActionListener ke cancelButton:

`cancelButton.addActionListener(e -> { ... });`

- Menambahkan *action listener* ke cancelButton menggunakan ekspresi lambda. Ketika tombol ini diklik:
 - Line 120: `new DashboardFrame(player1).setVisible(b:true);` - Membuat instance baru dari DashboardFrame (jendela dashboard utama), meneruskan objek player1, dan membuatnya terlihat.
 - Line 121: `dispose();` - Menutup dan membebaskan sumber daya dari jendela Player2LoginFrame saat ini.

```

124     gbc.gridx = 0;
125     gbc.gridy = 4;
126     gbc.gridwidth = 2;
127     gbc.insets = new Insets(top:0, left:0, bottom:0, right:0);
128     formPanel.add(buttonPanel, gbc);
129
130     // Menambahkan formPanel ke panel wrapper untuk penempatan di tengah frame
131     JPanel centerWrapperPanel = new JPanel(new GridBagLayout());
132     centerWrapperPanel.setBackground(new Color(r:30, g:30, b:30));
133     GridBagConstraints gbcWrapper = new GridBagConstraints();
134     gbcWrapper.gridx = 0;
135     gbcWrapper.gridy = 0;
136     gbcWrapper.weightx = 1.0;
137     gbcWrapper.weighty = 1.0;
138     gbcWrapper.anchor = GridBagConstraints.CENTER;
139     centerWrapperPanel.add(formPanel, gbcWrapper);
140     add(centerWrapperPanel, BorderLayout.CENTER);
141 }

```

Line 124: Mengatur gridx untuk gbc: gbc.gridx = 0;

- Mengatur posisi kolom (gridx) untuk komponen berikutnya (yang akan ditambahkan dengan gbc ini) menjadi kolom pertama (indeks 0) dalam formPanel.

Line 125: Mengatur gridy untuk gbc: gbc.gridy = 4;

- Mengatur posisi baris (gridy) untuk komponen berikutnya menjadi baris kelima (indeks 4) dalam formPanel, di bawah password field.

Line 126: Mengatur gridwidth untuk gbc: gbc.gridwidth = 2;

- Mengatur gridwidth untuk gbc menjadi 2. Ini berarti komponen ini akan membentang melintasi 2 kolom dalam grid. Ini berguna untuk menengahkan tombol login dan batal yang berada di satu panel (buttonPanel).

Line 127: Mengatur Inset gbc (ulang): gbc.insets = new Insets(top:0, left:0, bottom:0, right:0);

- Penting: Line ini menimpa pengaturan insets sebelumnya. Sekarang, komponen yang ditambahkan dengan gbc ini akan memiliki padding 0 piksel di semua sisi.

Line 128: Menambahkan buttonPanel ke formPanel: formPanel.add(buttonPanel, gbc);

- Menambahkan buttonPanel (yang berisi tombol login dan batal) ke formPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

Line 130: Komentar: // Menambahkan formPanel ke panel wrapper untuk penempatan di tengah frame - Komentar ini menjelaskan tujuan dari bagian kode berikutnya.

Line 131: Inisialisasi centerWrapperPanel: JPanel centerWrapperPanel = new JPanel(new GridBagLayout());

- Membuat objek JPanel baru bernama centerWrapperPanel. Panel ini berfungsi sebagai "pembungkus" untuk formPanel sehingga formPanel dapat ditempatkan dengan rapi di tengah frame utama menggunakan GridBagLayout.

Line 132: Mengatur Latar Belakang centerWrapperPanel:

```
centerWrapperPanel.setBackground(new Color(r:30, g:30, b:30));
```

- Mengatur warna latar belakang centerWrapperPanel menjadi abu-abu sangat gelap (Color(30, 30, 30)), sama dengan latar belakang utama frame.

Line 133: Inisialisasi GridBagConstraints untuk Wrapper: GridBagConstraints gbcWrapper = new GridBagConstraints();

- Membuat objek GridBagConstraints baru bernama gbcWrapper. Objek ini akan digunakan untuk menentukan bagaimana formPanel ditempatkan di dalam centerWrapperPanel.

Line 134: Mengatur gridx untuk gbcWrapper: gbcWrapper.gridx = 0;

- Mengatur posisi kolom (gridx) untuk komponen berikutnya (yaitu formPanel) menjadi kolom pertama (indeks 0) dalam centerWrapperPanel.

Line 135: Mengatur gridy untuk gbcWrapper: gbcWrapper.gridy = 0;

- Mengatur posisi baris (gridy) untuk komponen berikutnya menjadi baris pertama (indeks 0) dalam centerWrapperPanel.

Line 136: Mengatur weightx untuk gbcWrapper: gbcWrapper.weightx = 1.0;

- Mengatur weightx untuk gbcWrapper menjadi 1.0. Ini berarti formPanel akan mengambil semua ruang horizontal ekstra yang tersedia di kolomnya di dalam centerWrapperPanel.

Line 137: Mengatur weighty untuk gbcWrapper: gbcWrapper.weighty = 1.0;

- Mengatur weighty untuk gbcWrapper menjadi 1.0. Ini berarti formPanel akan mengambil semua ruang vertikal ekstra yang tersedia di barisnya di dalam centerWrapperPanel. Pengaturan weightx dan weighty ini, dikombinasikan dengan anchor di Line 138, adalah kunci untuk memusatkan komponen.

Line 138: Mengatur anchor untuk gbcWrapper: gbcWrapper.anchor = GridBagConstraints.CENTER;

- Mengatur anchor properti gbcWrapper menjadi CENTER. Karena weightx dan weighty diatur ke 1.0, ini memastikan formPanel akan diposisikan tepat di tengah centerWrapperPanel, memanfaatkan semua ruang ekstra.

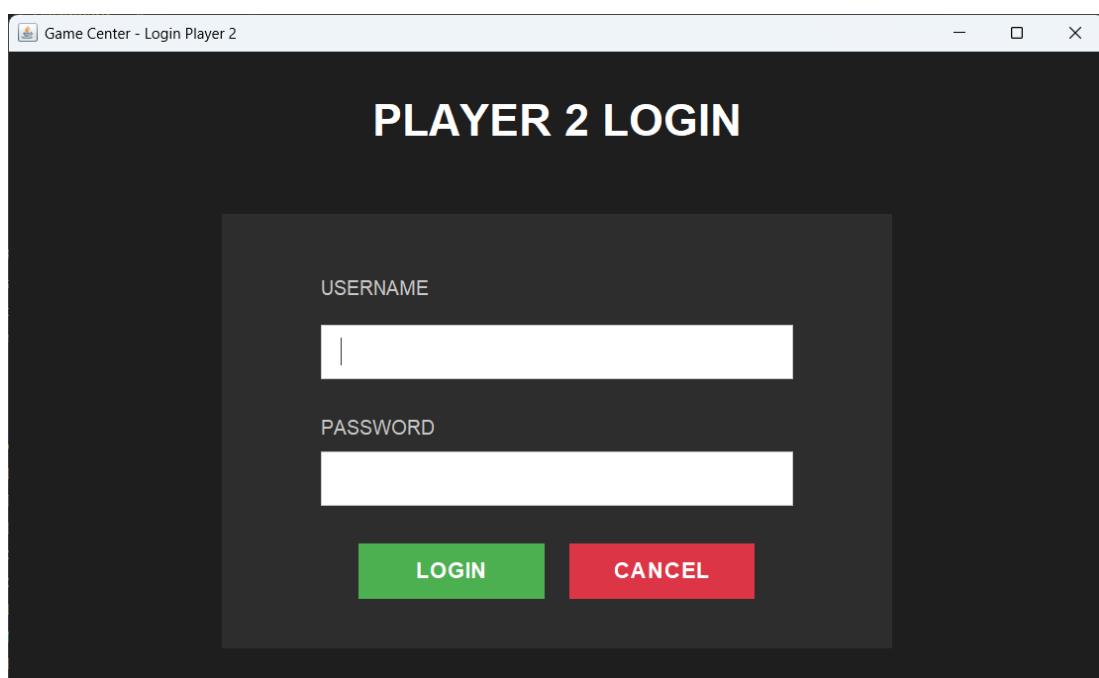
Line 139: Menambahkan formPanel ke centerWrapperPanel:
centerWrapperPanel.add(formPanel, gbcWrapper);

- Menambahkan formPanel ke centerWrapperPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbcWrapper).

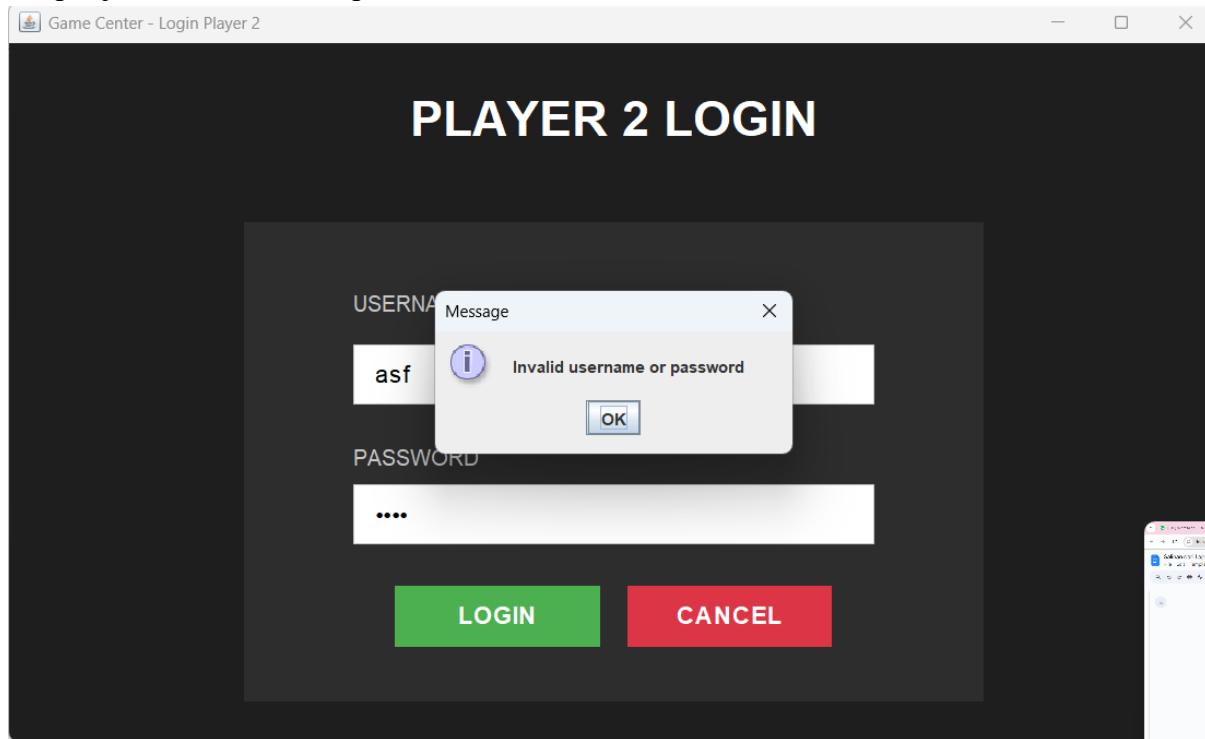
Line 140: Menambahkan centerWrapperPanel ke Frame Utama: add(centerWrapperPanel, BorderLayout.CENTER);

- Menambahkan centerWrapperPanel (yang sekarang berisi formPanel yang sudah terpusat) ke frame utama. Karena frame utama menggunakan BorderLayout, centerWrapperPanel ditempatkan di area CENTER, sehingga formulir login akan muncul di tengah jendela.

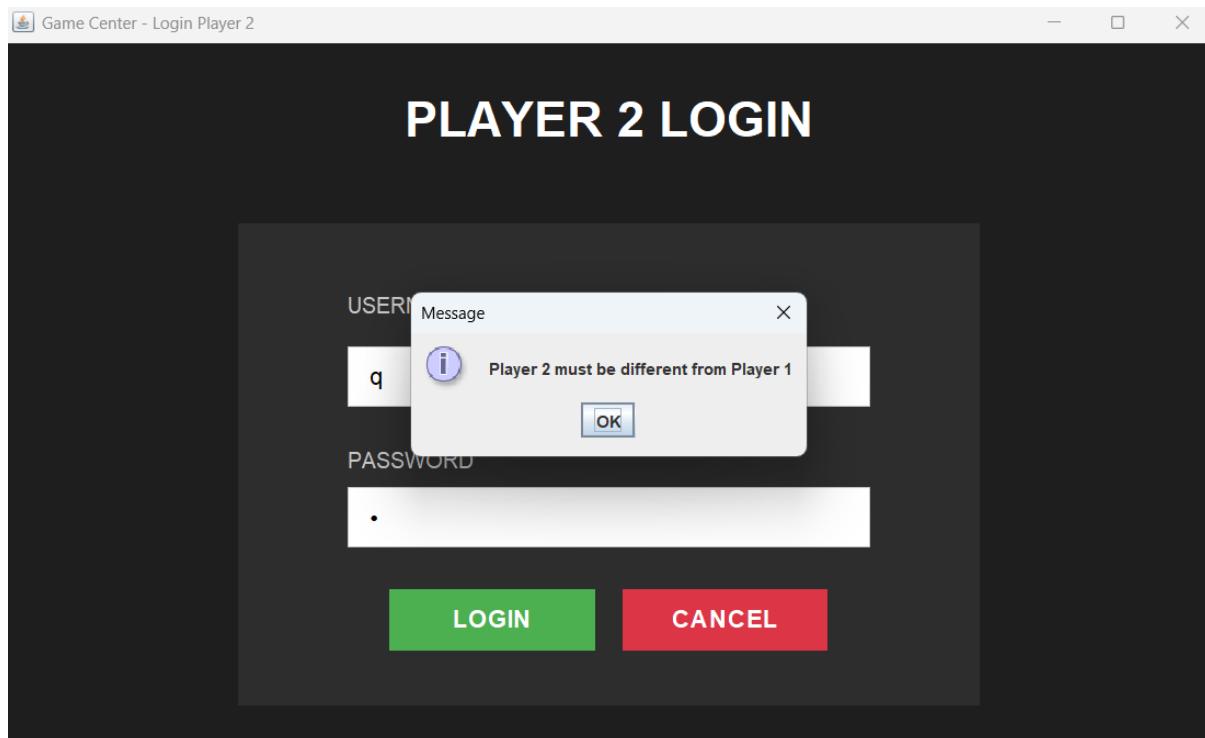
Output Player2Login :



Output jika username atau password tidak terdaftar :



Output jika login dengan akun user 1 :



RegisterFrame.java

```
1 package code;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.FocusAdapter;
7 import java.awt.event.FocusEvent;
8 import java.sql.*;
```

Line 3: Import Semua Kelas Swing: import javax.swing.*;

- Line ini mengimpor semua kelas dari package javax.swing. Package Swing menyediakan komponen GUI (Graphical User Interface) untuk aplikasi Java, seperti JFrame, JPanel, JButton, JLabel, JOptionPane, dan Timer.

Line 4: Import Semua Kelas AWT: import java.awt.*;

- Line ini mengimpor semua kelas dari package java.awt. AWT (Abstract Window Toolkit) adalah *toolkit* GUI asli Java. Beberapa kelas AWT masih sering digunakan bersama Swing, seperti Graphics, Color, Font, dan Dimension.

Line 5: Import Kelas ActionEvent: import java.awt.event.ActionEvent;

- Line ini mengimpor kelas ActionEvent secara spesifik dari package java.awt.event. ActionEvent digunakan untuk *event* yang dihasilkan oleh komponen AWT atau Swing, seperti ketika tombol diklik.

Line 6: Import Kelas FocusAdapter: import java.awt.event.FocusAdapter;

- Line ini mengimpor kelas FocusAdapter dari package java.awt.event. FocusAdapter adalah kelas *adapter* yang nyaman untuk membuat *listener* fokus. Ini menyediakan implementasi kosong dari semua metode FocusListener, sehingga Anda hanya perlu meng-override metode yang Anda butuhkan.

Line 7: Import Kelas FocusEvent: import java.awt.event.FocusEvent;

- Line ini mengimpor kelas FocusEvent secara spesifik dari package java.awt.event. FocusEvent adalah *event* yang dihasilkan ketika komponen mendapatkan atau kehilangan fokus.

Line 8: Import Semua Kelas SQL: import java.sql.*;

- Line ini mengimpor semua kelas dari package java.sql. Package ini menyediakan API Java Database Connectivity (JDBC) untuk berinteraksi dengan database, termasuk kelas seperti Connection, PreparedStatement, SQLException, dll. Ini mengindikasikan bahwa aplikasi ini akan berinteraksi dengan database (misalnya, untuk menyimpan atau mengambil data).

```
10  public class RegisterFrame extends JFrame {
11      private JTextField usernameField, nicknameField;
12      private JPasswordField passwordField;
13  
```

Line 10: Deklarasi Kelas RegisterFrame: public class RegisterFrame extends JFrame {

- Line ini mendeklarasikan kelas publik bernama RegisterFrame. Kata kunci extends JFrame menunjukkan bahwa RegisterFrame adalah *subclass* dari JFrame. Ini berarti RegisterFrame akan menjadi sebuah jendela aplikasi GUI, yang dirancang untuk proses registrasi pengguna.

Line 11: Deklarasi Variabel usernameField dan nicknameField: private JTextField usernameField, nicknameField;

- Line ini mendeklarasikan dua variabel *instance* privat, usernameField dan nicknameField, keduanya bertipe JTextField.
 - usernameField kemungkinan adalah komponen input teks tempat pengguna akan memasukkan nama pengguna yang mereka inginkan untuk login.
 - nicknameField kemungkinan adalah komponen input teks tempat pengguna akan memasukkan nama panggilan (nickname) yang akan ditampilkan dalam permainan atau profil.

Line 12: Deklarasi Variabel passwordField: private JPasswordField passwordField;

- Line ini mendeklarasikan variabel *instance* privat bernama passwordField dengan tipe JPasswordField. Ini adalah komponen input teks khusus yang dirancang untuk kata sandi, yang biasanya menyembunyikan karakter yang diketik untuk keamanan.

```

14  public RegisterFrame() {
15      setTitle(title:"Game Center - Register");
16      setSize(width:900, height:550);
17      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
18      setLocationRelativeTo(c:null);
19
20      getContentPane().setBackground(new Color(r:30, g:30, b:30));
21      setLayout(new BorderLayout());
22
23      JPanel topHeaderPanel = new JPanel();
24      topHeaderPanel.setBackground(new Color(r:30, g:30, b:30));
25      topHeaderPanel.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:10, right:0));
26
27      JLabel topTitleLabel = new JLabel(text:"Game Center");
28      topTitleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:36));
29      topTitleLabel.setForeground(Color.WHITE);
30      topHeaderPanel.add(topTitleLabel);
31      add(topHeaderPanel, BorderLayout.NORTH);
32
33      JPanel contentPanel = new JPanel(new GridBagLayout());
34      contentPanel.setBackground(new Color(r:30, g:30, b:30));
35
36      GridBagConstraints gbc = new GridBagConstraints();
37
38      gbc.insets = new Insets(top:10, left:100, bottom:10, right:100);
39
40      gbc.fill = GridBagConstraints.HORIZONTAL;
41
42
43      gbc.anchor = GridBagConstraints.CENTER;
44

```

Line 14: Konstruktor RegisterFrame: public RegisterFrame() {

- Ini adalah konstruktor untuk kelas RegisterFrame. Ketika objek RegisterFrame baru dibuat, kode di dalam blok ini akan dijalankan untuk menginisialisasi jendela pendaftaran.

Line 15: Mengatur Judul Jendela: setTitle(title:"Game Center - Register");

- Line ini mengatur judul jendela (frame) aplikasi menjadi "Game Center - Register". Judul ini akan muncul di bilah judul jendela.

Line 16: Mengatur Ukuran Jendela: setSize(width:900, height:550);

- Line ini mengatur ukuran jendela aplikasi. Lebarnya diatur menjadi 900 piksel dan tingginya menjadi 550 piksel.

Line 17: Mengatur Operasi Tutup Default:

```
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

- Line ini mengatur perilaku jendela saat tombol tutup (ikon 'X' di bilah judul) diklik. JFrame.DISPOSE_ON_CLOSE berarti jendela akan ditutup dan sumber dayanya dibebaskan, tetapi aplikasi Java secara keseluruhan mungkin tidak berakhir jika ada jendela lain yang masih terbuka.

Line 18: Memposisikan Jendela di Tengah: setLocationRelativeTo(c:null);

- Line ini memposisikan jendela di tengah layar. Ketika argumennya null, jendela akan berpusat pada layar.

Line 20: Mengatur Warna Latar Belakang Konten Pane:

```
getContentPane().setBackground(new Color(r:30, g:30, b:30));
```

- Line ini mendapatkan *content pane* dari JFrame (area tempat komponen GUI akan ditambahkan) dan mengatur warna latar belakangnya menjadi abu-abu sangat gelap (new Color(30, 30, 30)).

Line 21: Mengatur Layout Manager Utama: setLayout(new BorderLayout());

- Line ini mengatur *layout manager* untuk JFrame ini menjadi BorderLayout. BorderLayout membagi ruang menjadi lima area: NORTH (atas), SOUTH (bawah), EAST (kanan), WEST (kiri), dan CENTER (tengah).

Line 23: Inisialisasi topHeaderPanel: JPanel topHeaderPanel = new JPanel();

- Membuat objek JPanel baru bernama topHeaderPanel. Panel ini akan digunakan untuk menampung elemen judul di bagian atas frame.

Line 24: Mengatur Latar Belakang topHeaderPanel: topHeaderPanel.setBackground(new Color(r:30, g:30, b:30));

- Mengatur warna latar belakang topHeaderPanel menjadi abu-abu sangat gelap (Color(30, 30, 30)), sama dengan latar belakang frame.

Line 25: Mengatur Border topHeaderPanel:

```
topHeaderPanel.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:10, right:0));
```

- Mengatur border kosong (padding) untuk topHeaderPanel. Ini menambahkan ruang kosong 20 piksel di atas dan 10 piksel di bawah, serta 0 di kiri dan kanan.

Line 26: Inisialisasi topTitleLabel: JLabel topTitleLabel = new JLabel(text:"Game Center");

- Membuat objek JLabel baru bernama topTitleLabel dengan teks "Game Center". Ini akan berfungsi sebagai judul utama di bagian atas jendela.

Line 27: Mengatur Font topTitleLabel: topTitleLabel.setFont(new Font(name:"Arial", font:Font.BOLD, size:36));

- Mengatur font untuk topTitleLabel. Fontnya adalah "Arial", gayanya tebal (Font.BOLD), dan ukurannya 36.

Line 28: Mengatur Warna Teks topTitleLabel: topTitleLabel.setForeground(Color.WHITE);

- Mengatur warna teks topTitleLabel menjadi putih.

Line 30: Menambahkan topTitleLabel ke topHeaderPanel:

```
topHeaderPanel.add(topTitleLabel);
```

- Menambahkan topTitleLabel sebagai komponen ke topHeaderPanel.

Line 31: Menambahkan topHeaderPanel ke Frame Utama: add(topHeaderPanel, BorderLayout.NORTH);

- Menambahkan topHeaderPanel (yang berisi judul "Game Center") ke frame utama. Karena frame utama menggunakan BorderLayout, topHeaderPanel ditempatkan di area NORTH (atas).

Line 33: Inisialisasi contentPanel: JPanel contentPanel = new JPanel(new GridBagLayout());

- Membuat objek JPanel baru bernama contentPanel. Panel ini akan menjadi wadah utama untuk formulir pendaftaran, menggunakan GridBagLayout untuk penempatan komponen yang fleksibel.

Line 34: Mengatur Latar Belakang contentPanel: contentPanel.setBackground(new Color(r:30, g:30, b:30));

- Mengatur warna latar belakang contentPanel menjadi abu-abu sangat gelap (Color(30, 30, 30)), sama dengan latar belakang frame.

Line 36: Inisialisasi GridBagConstraints: GridBagConstraints gbc = new GridBagConstraints();

- Membuat objek GridBagConstraints baru bernama gbc. Objek ini akan digunakan untuk menentukan bagaimana komponen akan ditempatkan dan perlakunya dalam GridBagLayout.

Line 38: Mengatur Inset gbc: gbc.insets = new Insets(top:10, left:100, bottom:10, right:100);

- Mengatur insets (padding eksternal) untuk komponen yang akan menggunakan gbc. Ini menambahkan ruang kosong 10 piksel di atas dan bawah, serta 100 piksel di kiri dan kanan setiap komponen.

Line 40: Mengatur fill gbc: gbc.fill = GridBagConstraints.HORIZONTAL;

- Mengatur fill properti gbc menjadi HORIZONTAL. Ini berarti komponen akan mengisi ruang horizontal yang tersedia di sel grid-nya jika ada ruang kosong.

Line 43: Mengatur anchor gbc: gbc.anchor = GridBagConstraints.CENTER;

- Mengatur anchor properti gbc menjadi CENTER. Ini berarti jika komponen tidak mengisi seluruh ruang yang tersedia di sel grid-nya, komponen akan diposisikan di tengah sel tersebut.

```

45     // --- Username Field ---
46     usernameField = new JTextField(text:"Username", columns:25);
47     usernameField.setFont(new Font(name:"Arial", Font.PLAIN, size:18));
48     usernameField.setBackground(Color.WHITE);
49     usernameField.setForeground(Color.GRAY);
50     usernameField.setCaretColor(Color.BLACK);
51     usernameField.setBorder(BorderFactory.createCompoundBorder(
52         BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1),
53         BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15)
54     ));
55
56     usernameField.setPreferredSize(new Dimension(width:300, height:45));
57
58     usernameField.addFocusListener(new FocusAdapter() {
59         @Override
60         public void focusGained(FocusEvent e) {
61             if (usernameField.getText().equals(anObject:"Username")) {
62                 usernameField.setText(t:@"");
63                 usernameField.setForeground(Color.BLACK);
64             }
65         }
66         @Override
67         public void focusLost(FocusEvent e) {
68             if (usernameField.getText().isEmpty()) {
69                 usernameField.setText(t:"Username");
70                 usernameField.setForeground(Color.GRAY);
71             }
72         }
73     });
74     gbc.gridx = 0;
75     gbc.gridy = 0;
76
77     gbc.weightx = 1.0;
78     contentPanel.add(usernameField, gbc);

```

Line 46: Inisialisasi usernameField: usernameField = new JTextField(text:"Username", columns:25);

- Membuat objek JTextField baru bernama usernameField. Parameter text:"Username" mengatur teks *placeholder* awal di dalam field, dan columns:25 memberikan lebar preferensi 25 karakter.

Line 47: Mengatur Font usernameField: usernameField.setFont(new Font(name:"Arial", font:Font.PLAIN, size:18));

- Mengatur font untuk teks di usernameField. Fontnya "Arial", gaya PLAIN (normal), dan ukurannya 18.

Line 48: Mengatur Latar Belakang usernameField:
usernameField.setBackground(Color.WHITE);

- Mengatur warna latar belakang usernameField menjadi putih.

Line 49: Mengatur Warna Teks usernameField: usernameField.setForeground(Color.GRAY);

- Mengatur warna teks default di usernameField menjadi abu-abu. Ini akan digunakan untuk teks *placeholder* "Username".

Line 50: Mengatur Warna Caret usernameField:

```
usernameField.setCaretColor(Color.BLACK);
```

- Mengatur warna *caret* (kursor yang berkedip) di usernameField menjadi hitam.

Line 51-54: Mengatur Border usernameField:

```
usernameField.setBorder(BorderFactory.createCompoundBorder(...));
```

- Mengatur border gabungan (CompoundBorder) untuk usernameField. Ini menggabungkan dua jenis border:
 - Line 52: BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1): Membuat border garis tipis (1 piksel) berwarna abu-abu muda di sekeliling field.
 - Line 53: BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15): Membuat border kosong (padding internal) di dalam border garis, menambahkan ruang 10 piksel di atas dan bawah, serta 15 piksel di kiri dan kanan teks. Ini memberikan jarak antara teks dan batas field.

Line 56: Mengatur Ukuran Preferensi usernameField: usernameField.setPreferredSize(new Dimension(width:300, height:45));

- Mengatur ukuran yang disukai (preferensi) untuk usernameField menjadi lebar 300 piksel dan tinggi 45 piksel.

Line 58-73: Menambahkan FocusListener ke usernameField:

```
usernameField.addFocusListener(new FocusAdapter() { ... });
```

- Menambahkan FocusListener ke usernameField menggunakan FocusAdapter. Ini menangani perubahan fokus (saat field dipilih atau tidak dipilih).
 - Line 60-64: @Override public void focusGained(FocusEvent e) { ... } - Metode ini dipanggil ketika usernameField mendapatkan fokus (pengguna mengklik atau menab ke field).
 - Line 61: if (usernameField.getText().equals("Username")) { - Memeriksa apakah teks saat ini di field adalah "Username" (teks *placeholder*).
 - Line 62: usernameField.setText(""); - Jika itu adalah *placeholder*, teks dihapus.
 - Line 63: usernameField.setForeground(Color.BLACK); - Warna teks diatur menjadi hitam agar input pengguna terlihat jelas.
 - Line 67-71: @Override public void focusLost(FocusEvent e) { ... } - Metode ini dipanggil ketika usernameField kehilangan fokus.

- Line 68: if (usernameField.getText().isEmpty()) { - Memeriksa apakah field kosong setelah kehilangan fokus.
- Line 69: usernameField.setText("Username"); - Jika kosong, teks *placeholder* "Username" ditambahkan kembali.
- Line 70: usernameField.setForeground(Color.GRAY); - Warna teks diatur kembali menjadi abu-abu untuk *placeholder*.

Line 74: Mengatur gridx untuk gbc: gbc.gridx = 0;

- Mengatur posisi kolom (gridx) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi kolom pertama (indeks 0).

Line 75: Mengatur gridy untuk gbc: gbc.gridy = 0;

- Mengatur posisi baris (gridy) untuk komponen berikutnya menjadi baris pertama (indeks 0).

Line 77: Mengatur weightx untuk gbc: gbc.weightx = 1.0;

- Mengatur weightx untuk gbc menjadi 1.0. Ini berarti komponen ini akan mengambil semua ruang horizontal ekstra yang tersedia di kolomnya.

Line 78: Menambahkan usernameField ke contentPanel: contentPanel.add(usernameField, gbc);

- Menambahkan usernameField ke contentPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

```

80     // --- Password Field ---
81     passwordField = new JPasswordField(text:"Password", columns:25);
82     passwordField.setFont(new Font(name:"Arial", Font.PLAIN, size:18));
83     passwordField.setBackground(Color.WHITE);
84     passwordField.setForeground(Color.GRAY);
85     passwordField.setCaretColor(Color.BLACK);
86     passwordField.setBorder(BorderFactory.createCompoundBorder(
87         BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1),
88         BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15)
89     ));
90     // --- PENTING: setPreferredSize() untuk Password Field ---
91     passwordField.setPreferredSize(new Dimension(width:300, height:45));
92
93     passwordField.setEchoChar((char)0);
94     passwordField.addFocusListener(new FocusAdapter() {
95         @Override
96         public void focusGained(FocusEvent e) {
97             if (new String(passwordField.getPassword()).equals(anObject:"Password")) {
98                 passwordField.setText(t:"");
99                 passwordField.setEchoChar(c:'*');
100                passwordField.setForeground(Color.BLACK);
101            }
102        }
103        @Override
104        public void focusLost(FocusEvent e) {
105            if (new String(passwordField.getPassword()).isEmpty()) {
106                passwordField.setText(t:"Password");
107                passwordField.setEchoChar((char)0);
108                passwordField.setForeground(Color.GRAY);
109            }
110        }
111    });
112    gbc.gridx = 1;
113    // mengatur jarak antara field password dan username
114    gbc.insets = new Insets(top:15, left:100, bottom:15, right:100);
115    contentPanel.add(passwordField, gbc);

```

Line 81: Inisialisasi passwordField: passwordField = new JPasswordField(text:"Password", columns:25);

- Membuat objek JPasswordField baru bernama passwordField. Parameter text:"Password" mengatur teks *placeholder* awal di dalam field, dan columns:25 memberikan lebar preferensi 25 karakter.

Line 82: Mengatur Font passwordField: passwordField.setFont(new Font(name:"Arial", font:Font.PLAIN, size:18));

- Mengatur font untuk teks di passwordField. Fontnya "Arial", gaya PLAIN (normal), dan ukurannya 18.

Line 83: Mengatur Latar Belakang passwordField:
passwordField.setBackground(Color.WHITE);

- Mengatur warna latar belakang passwordField menjadi putih.

Line 84: Mengatur Warna Teks passwordField: passwordField.setForeground(Color.GRAY);

- Mengatur warna teks default di passwordField menjadi abu-abu. Ini akan digunakan untuk teks *placeholder* "Password".

Line 85: Mengatur Warna Caret passwordField:
`passwordField.setCaretColor(Color.BLACK);`

Mengatur warna *caret* (kursor yang berkedip) di passwordField menjadi hitam.

Line 86-89: Mengatur Border passwordField:
`passwordField.setBorder(BorderFactory.createCompoundBorder(...));`

- Mengatur border gabungan (CompoundBorder) untuk passwordField. Ini menggabungkan dua jenis border:
 - Line 87: `BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1)`: Membuat border garis tipis (1 piksel) berwarna abu-abu muda di sekeliling field.
 - Line 88: `BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15)`: Membuat border kosong (padding internal) di dalam border garis, menambahkan ruang 10 piksel di atas dan bawah, serta 15 piksel di kiri dan kanan teks. Ini memberikan jarak antara teks dan batas field.

Line 91: Mengatur Ukuran Preferensi passwordField: `passwordField.setPreferredSize(new Dimension(width:300, height:45));`

- Mengatur ukuran yang disukai (preferensi) untuk passwordField menjadi lebar 300 piksel dan tinggi 45 piksel.

Line 93: Mengatur Echo Character passwordField: `passwordField.setEchoChar(char:0);`

- Mengatur karakter gema (karakter yang ditampilkan saat mengetik) menjadi 0. Ini kemungkinan dilakukan untuk sementara waktu agar teks *placeholder* "Password" terlihat, sebelum diganti dengan '*' atau karakter lain.

Line 94-110: Menambahkan FocusListener ke passwordField:
`passwordField.addFocusListener(new FocusAdapter() { ... });`

- Menambahkan FocusListener ke passwordField menggunakan FocusAdapter. Ini menangani perubahan fokus (saat field dipilih atau tidak dipilih), serupa dengan usernameField.
 - Line 96-101: `@Override public void focusGained(FocusEvent e) { ... }` - Metode ini dipanggil ketika passwordField mendapatkan fokus.

- Line 97: if (new String(passwordField.getPassword()).equals("Password")) { - Memeriksa apakah teks saat ini di field adalah "Password" (teks *placeholder*). passwordField.getPassword() mengembalikan char[], jadi perlu diubah ke String untuk perbandingan.
- Line 98: passwordField.setText(""); - Jika itu adalah *placeholder*, teks dihapus.
- Line 99: passwordField.setEchoChar('*'); - Karakter gema diatur menjadi *, sehingga karakter yang diketik akan disembunyikan.
- Line 100: passwordField.setForeground(Color.BLACK); - Warna teks diatur menjadi hitam agar input pengguna terlihat jelas.
- Line 104-108: @Override public void focusLost(FocusEvent e) { ... } - Metode ini dipanggil ketika passwordField kehilangan fokus.
 - Line 105: if (new String(passwordField.getPassword()).isEmpty()) { - Memeriksa apakah field kosong setelah kehilangan fokus.
 - Line 106: passwordField.setText("Password"); - Jika kosong, teks *placeholder* "Password" ditambahkan kembali.
 - Line 107: passwordField.setEchoChar(char:0); - Karakter gema diatur kembali ke 0, sehingga *placeholder* "Password" dapat terlihat.
 - Line 108: passwordField.setForeground(Color.GRAY); - Warna teks diatur kembali menjadi abu-abu untuk *placeholder*.

Line 112: Mengatur gridy untuk gbc: gbc.gridx = 1;

- Mengatur posisi baris (gridy) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi baris kedua (indeks 1), di bawah usernameField.

Line 114: Mengatur Inset gbc (ulang): gbc.insets = new Insets(top:15, left:100, bottom:15, right:100);

- Penting: Line ini menimpa pengaturan insets sebelumnya. Sekarang, komponen yang ditambahkan dengan gbc ini akan memiliki padding 15 piksel di atas dan bawah, serta 100 piksel di kiri dan kanan. Ini memberikan jarak vertikal yang spesifik antara usernameField dan passwordField.

Line 115: Menambahkan passwordField ke contentPanel: contentPanel.add(passwordField, gbc);

- Menambahkan passwordField ke contentPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

```

118     nicknameField = new JTextField(text:"Nickname", columns:25);
119     nicknameField.setFont(new Font(name:"Arial", Font.PLAIN, size:18));
120     nicknameField.setBackground(Color.WHITE);
121     nicknameField.setForeground(Color.GRAY);
122     nicknameField.setCaretColor(Color.BLACK);
123     nicknameField.setBorder(BorderFactory.createCompoundBorder(
124         BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1),
125         BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15)
126     ));
127     //setPreferredSize() untuk Nickname Field ---
128     nicknameField.setPreferredSize(new Dimension(width:300, height:45));
129
130     nicknameField.addFocusListener(new FocusAdapter() {
131         @Override
132         public void focusGained(FocusEvent e) {
133             if (nicknameField.getText().equals(anObject:"Nickname")) {
134                 nicknameField.setText(t:@"");
135                 nicknameField.setForeground(Color.BLACK);
136             }
137         }
138         @Override
139         public void focusLost(FocusEvent e) {
140             if (nicknameField.getText().isEmpty()) {
141                 nicknameField.setText(t:"Nickname");
142                 nicknameField.setForeground(Color.GRAY);
143             }
144         }
145     });
146     gbc.gridx = 2;
147     // mengatur jarak antara field nickname dan password, serta jarak di bawah field terakhir
148     gbc.insets = new Insets(top:15, left:100, bottom:30, right:100);
149     contentPanel.add(nicknameField, gbc);

```

Line 118: Inisialisasi nicknameField: nicknameField = new JTextField(text:"Nickname", columns:25);

- Membuat objek JTextField baru bernama nicknameField. Parameter text:"Nickname" mengatur teks *placeholder* awal, dan columns:25 memberikan lebar preferensi 25 karakter.

Line 119: Mengatur Font nicknameField: nicknameField.setFont(new Font(name:"Arial", font:Font.PLAIN, size:18));

- Mengatur font untuk teks di nicknameField. Fontnya "Arial", gaya PLAIN (normal), dan ukurannya 18.

Line 120: Mengatur Latar Belakang nicknameField:
nicknameField.setBackground(Color.WHITE);

- Mengatur warna latar belakang nicknameField menjadi putih.

Line 121: Mengatur Warna Teks nicknameField:
nicknameField.setForeground(Color.GRAY);

- Mengatur warna teks default di nicknameField menjadi abu-abu untuk teks *placeholder* "Nickname".

Line 122: Mengatur Warna Caret nicknameField:
nicknameField.setCaretColor(Color.BLACK);

- Mengatur warna *caret* (kursor yang berkedip) di nicknameField menjadi hitam.

Line 123-126: Mengatur Border nicknameField:
nicknameField.setBorder(BorderFactory.createCompoundBorder(...));

- Mengatur border gabungan (CompoundBorder) untuk nicknameField. Ini menggabungkan:
 - Line 124: BorderFactory.createLineBorder(new Color(r:200, g:200, b:200), thickness:1): Border garis tipis (1 piksel) berwarna abu-abu muda di sekeliling field.
 - Line 125: BorderFactory.createEmptyBorder(top:10, left:15, bottom:10, right:15): Padding internal yang menambahkan ruang 10 piksel di atas dan bawah, serta 15 piksel di kiri dan kanan teks di dalam field.

Line 128: Mengatur Ukuran Preferensi nicknameField: nicknameField.setPreferredSize(new Dimension(width:300, height:45));

- Mengatur ukuran yang disukai (preferensi) untuk nicknameField menjadi lebar 300 piksel dan tinggi 45 piksel.

Line 130-145: Menambahkan FocusListener ke nicknameField:
nicknameField.addFocusListener(new FocusAdapter() { ... });

- Menambahkan FocusListener ke nicknameField menggunakan FocusAdapter. Ini menangani perubahan fokus, mirip dengan usernameField dan passwordField.
 - Line 132-136: @Override public void focusGained(FocusEvent e) { ... } - Dipanggil saat nicknameField mendapatkan fokus.
 - Line 133: if (nicknameField.getText().equals("Nickname")) { - Memeriksa apakah teks saat ini adalah *placeholder*.
 - Line 134: nicknameField.setText(""); - Jika *placeholder*, teks dihapus.
 - Line 135: nicknameField.setForeground(Color.BLACK); - Warna teks diatur menjadi hitam.
 - Line 139-143: @Override public void focusLost(FocusEvent e) { ... } - Dipanggil saat nicknameField kehilangan fokus.
 - Line 140: if (nicknameField.getText().isEmpty()) { - Memeriksa apakah field kosong.
 - Line 141: nicknameField.setText("Nickname"); - Jika kosong, *placeholder* "Nickname" ditambahkan kembali.
 - Line 142: nicknameField.setForeground(Color.GRAY); - Warna teks diatur kembali menjadi abu-abu.

Line 146: Mengatur gridy untuk gbc: gbc.gridx = 2;

- Mengatur posisi baris (gridy) untuk komponen berikutnya yang akan ditambahkan dengan gbc ini menjadi baris ketiga (indeks 2), di bawah password field.

Line 148: Mengatur Inset gbc (ulang): gbc.insets = new Insets(top:15, left:100, bottom:30, right:100);

- Penting: Line ini menimpa pengaturan insets sebelumnya. Sekarang, komponen yang ditambahkan dengan gbc ini akan memiliki padding 15 piksel di atas, 100 piksel di kiri dan kanan, serta 30 piksel di bawah. Padding bawah 30 piksel ini akan memberikan jarak yang cukup antara nicknameField dan tombol register/cancel yang akan datang.

Line 149: Menambahkan nicknameField ke contentPanel: contentPanel.add(nicknameField, gbc);

- Menambahkan nicknameField ke contentPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

```

152     // --- Panel Tombol Register/Back
153     JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, hgap:20, vgap:0));
154     buttonPanel.setBackground(new Color(r:30, g:30, b:30));
155
156     JButton registerButton = new JButton(text:"Register");
157     registerButton.setFont(new Font(name:"Arial", Font.BOLD, size:18));
158     registerButton.setBackground(new Color(r:76, g:175, b:80));
159     registerButton.setForeground(Color.WHITE);
160     registerButton.setFocusPainted(b:false);
161     registerButton.setBorderPainted(b:false);
162     registerButton.setPreferredSize(new Dimension(width:150, height:45));
163     buttonPanel.add(registerButton);
164     registerButton.addActionListener(this::performRegister);
165
166     JButton backButton = new JButton(text:"Back");
167     backButton.setFont(new Font(name:"Arial", Font.BOLD, size:18));
168     backButton.setBackground(new Color(r:255, g:165, b:0));
169     backButton.setForeground(Color.WHITE);
170     backButton.setFocusPainted(b:false);
171     backButton.setBorderPainted(b:false);
172     backButton.setPreferredSize(new Dimension(width:150, height:45));
173     buttonPanel.add(backButton);
174     backButton.addActionListener(e -> {
175         new LoginFrame().setVisible(b:true);
176         dispose();
177     });
178
179     gbc.gridx = 0;
180     gbc.gridy = 3;
181     gbc.gridwidth = 1;
182     gbc.insets = new Insets(top:0, left:0, bottom:0, right:0);
183     contentPanel.add(buttonPanel, gbc);
184
185     add(contentPanel, BorderLayout.CENTER);
186 }
```

Line 153: Inisialisasi buttonPanel: JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, hgap:20, vgap:0));

- Membuat objek JPanel baru bernama buttonPanel. Panel ini akan menampung tombol-tombol. Ia menggunakan FlowLayout sebagai *layout manager* yang menata komponen dalam satu baris. FlowLayout.CENTER menengahkan komponen secara horizontal, hgap:20 memberikan celah horizontal 20 piksel antar komponen, dan vgap:0 berarti tidak ada celah vertikal antar baris (karena hanya ada satu baris).

Line 154: Mengatur Latar Belakang buttonPanel: buttonPanel.setBackground(new Color(r:30, g:30, b:30));

- Mengatur warna latar belakang buttonPanel menjadi abu-abu sangat gelap (Color(30, 30, 30)), sama dengan latar belakang frame utama.

Line 156: Inisialisasi registerButton: JButton registerButton = new JButton(text:"Register");

- Membuat objek JButton baru bernama registerButton dengan teks "Register".

Line 157: Mengatur Font registerButton: registerButton.setFont(new Font(name:"Arial", font:Font.BOLD, size:18));

- Mengatur font untuk tombol register. Fontnya "Arial", gaya BOLD (tebal), dan ukurannya 18.

Line 158: Mengatur Latar Belakang registerButton: registerButton.setBackground(new Color(r:76, g:175, b:80));

- Mengatur warna latar belakang tombol register menjadi hijau gelap (Color(76, 175, 80)).

Line 159: Mengatur Warna Teks registerButton:
registerButton.setForeground(Color.WHITE);

- Mengatur warna teks tombol register menjadi putih.

Line 160: Mengatur focusPainted registerButton: registerButton.setFocusPainted(b:false);

- Mengatur focusPainted menjadi false. Ini berarti tombol tidak akan menggambar kotak fokus di sekelilingnya saat mendapatkan fokus (misalnya, saat di-tab).

Line 161: Mengatur borderPainted registerButton: registerButton.setBorderPainted(b:false);

- Mengatur borderPainted menjadi false. Ini berarti tombol tidak akan menggambar batas default-nya.

Line 162: Mengatur Ukuran Preferensi registerButton: registerButton.setPreferredSize(new Dimension(width:150, height:45));

- Mengatur ukuran yang disukai (preferensi) untuk tombol register menjadi lebar 150 piksel dan tinggi 45 piksel.

Line 163: Menambahkan registerButton ke buttonPanel: buttonPanel.add(registerButton);

- Menambahkan registerButton sebagai komponen ke buttonPanel.

Line 164: Menambahkan ActionListener ke registerButton:

```
registerButton.addActionListener(this::performRegister);
```

- Menambahkan *action listener* ke registerButton. Ketika tombol ini diklik, metode performRegister dari kelas ini akan dipanggil. this::performRegister adalah referensi metode yang lebih ringkas.

Line 166: Inisialisasi backButton: JButton backButton = new JButton(text:"Back");

- Membuat objek JButton baru bernama backButton dengan teks "Back".

Line 167: Mengatur Font backButton: backButton.setFont(new Font(name:"Arial", font:Font.BOLD, size:18));

- Mengatur font untuk tombol kembali. Fontnya "Arial", gaya BOLD (tebal), dan ukurannya 18.

Line 168: Mengatur Latar Belakang backButton: backButton.setBackground(new Color(r:255, g:165, b:0));

- Mengatur warna latar belakang tombol kembali menjadi oranye (Color(255, 165, 0)).

Line 169: Mengatur Warna Teks backButton: backButton.setForeground(Color.WHITE);

- Mengatur warna teks tombol kembali menjadi putih.

Line 170: Mengatur focusPainted backButton: backButton.setFocusPainted(b:false);

- Mengatur focusPainted menjadi false untuk tombol kembali.

Line 171: Mengatur borderPainted backButton: backButton.setBorderPainted(b:false);

- Mengatur borderPainted menjadi false untuk tombol kembali.

Line 172: Mengatur Ukuran Preferensi backButton: backButton.setPreferredSize(new Dimension(width:150, height:45));

- Mengatur ukuran yang disukai (preferensi) untuk tombol kembali menjadi lebar 150 piksel dan tinggi 45 piksel.

Line 173: Menambahkan backButton ke buttonPanel: buttonPanel.add(backButton);

- Menambahkan backButton sebagai komponen ke buttonPanel.

Line 174-177: Menambahkan ActionListener ke backButton:

```
backButton.addActionListener(e -> { ... });
```

- Menambahkan *action listener* ke backButton menggunakan ekspresi lambda. Ketika tombol ini diklik:
 - Line 175: new LoginFrame().setVisible(b:true); - Membuat instance baru dari LoginFrame dan membuatnya terlihat (kembali ke layar login).
 - Line 176: dispose(); - Menutup dan membebaskan sumber daya dari jendela RegisterFrame saat ini.

Line 179: Mengatur gridx untuk gbc: gbc.gridx = 0;

- Mengatur posisi kolom (gridx) untuk buttonPanel menjadi kolom pertama (indeks 0) dalam contentPanel.

Line 180: Mengatur gridy untuk gbc: gbc.gridy = 3;

- Mengatur posisi baris (gridy) untuk buttonPanel menjadi baris keempat (indeks 3) dalam contentPanel, di bawah nicknameField.

Line 181: Mengatur gridwidth untuk gbc: gbc.gridwidth = 1;

- Mengatur gridwidth untuk gbc menjadi 1. Ini berarti buttonPanel akan menempati satu kolom dalam grid.

Line 182: Mengatur Inset gbc (ulang): gbc.insets = new Insets(top:0, left:0, bottom:0, right:0);

- Penting: Line ini menimpa pengaturan insets sebelumnya. Sekarang, komponen yang ditambahkan dengan gbc ini akan memiliki padding 0 piksel di semua sisi.

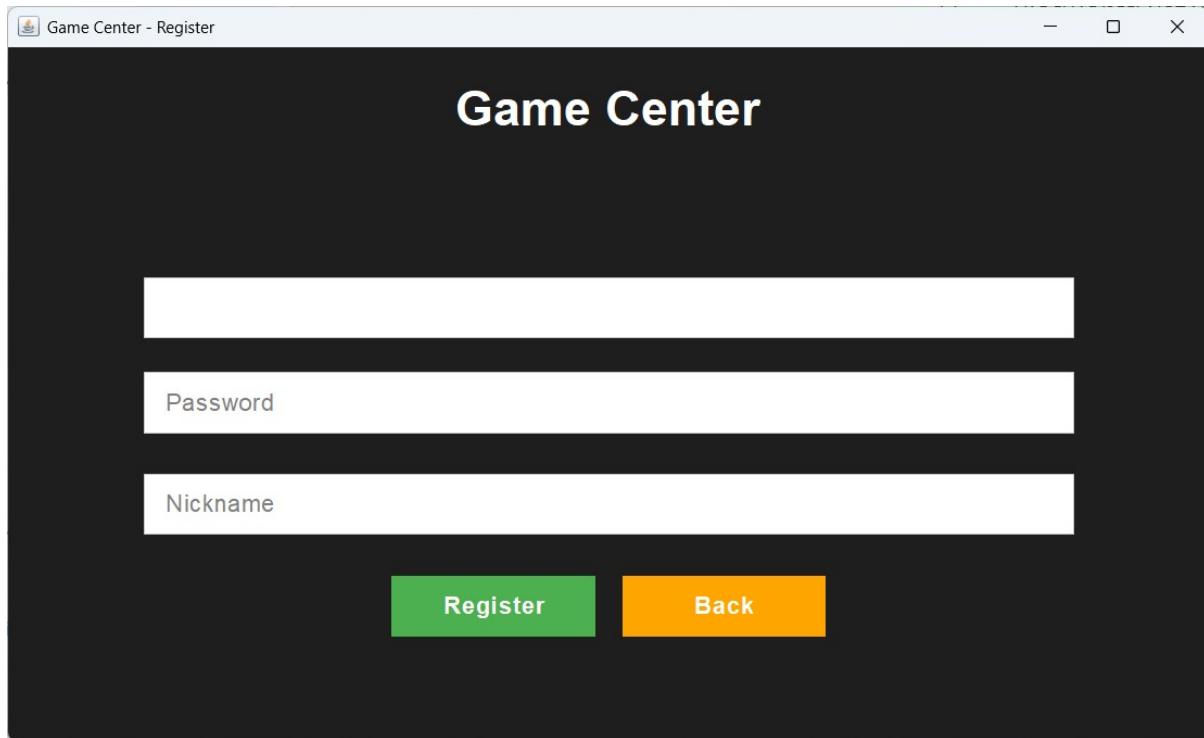
Line 183: Menambahkan buttonPanel ke contentPanel: contentPanel.add(buttonPanel, gbc);

- Menambahkan buttonPanel ke contentPanel menggunakan GridBagConstraints yang telah dikonfigurasi (gbc).

Line 185: Menambahkan contentPanel ke Frame Utama: add(contentPanel, BorderLayout.CENTER);

- Menambahkan contentPanel (yang berisi semua field input dan tombol) ke frame utama. Karena frame utama menggunakan BorderLayout, contentPanel ditempatkan di area CENTER, sehingga formulir pendaftaran akan muncul di tengah jendela.

Output tampilan Registrasi :



```
188     private void performRegister(ActionEvent e) {  
189         String username = usernameField.getText().trim();  
190         if (username.equals("Username")) username = "";  
191  
192         String password = new String(passwordField.getPassword()).trim();  
193         if (password.equals("Password")) password = "";  
194  
195         String nickname = nicknameField.getText().trim();  
196         if (nickname.equals("Nickname")) nickname = "";  
197  
198         if (username.isEmpty() || password.isEmpty() || nickname.isEmpty()) {  
199             JOptionPane.showMessageDialog(this, message:"Isi semua kolom!!");  
200             return;  
201         }  
202     }
```

Line 188: Deklarasi Metode performRegister(): private void performRegister(ActionEvent e)
{

- Mendefinisikan metode privat bernama performRegister. Metode ini akan dipanggil ketika tombol register diklik (menerima ActionEvent sebagai parameter), dan bertanggung jawab untuk memproses data pendaftaran.

Line 189: Mengambil Username: String username = usernameField.getText().trim();

- Mengambil teks dari usernameField (komponen input username). .getText() mendapatkan teksnya, dan .trim() menghapus spasi kosong di awal dan akhir string. Hasilnya disimpan dalam variabel username.

Line 190: Menangani Placeholder Username: if (username.equals("Username"))
username = "";

- Memeriksa apakah username yang diambil masih berupa teks *placeholder* "Username". Jika iya, username diatur ulang menjadi string kosong "". Ini mencegah *placeholder* disimpan sebagai username sebenarnya.

Line 192: Mengambil Password: String password = new String(passwordField.getPassword()).trim();

- Mengambil teks dari passwordField (komponen input password). passwordField.getPassword() mengembalikan char[] untuk alasan keamanan, jadi diubah menjadi String baru. .trim() juga menghapus spasi kosong. Hasilnya disimpan dalam variabel password.

Line 193: Menangani Placeholder Password: if (password.equals(anObject:"Password")) password = "";

- Memeriksa apakah password yang diambil masih berupa teks *placeholder* "Password". Jika iya, password diatur ulang menjadi string kosong "".

Line 195: Mengambil Nickname: String nickname = nicknameField.getText().trim();

- Mengambil teks dari nicknameField (komponen input nickname). .getText() mendapatkan teksnya, dan .trim() menghapus spasi kosong. Hasilnya disimpan dalam variabel nickname.

Line 196: Menangani Placeholder Nickname: if (nickname.equals(anObject:"Nickname")) nickname = "";

- Memeriksa apakah nickname yang diambil masih berupa teks *placeholder* "Nickname". Jika iya, nickname diatur ulang menjadi string kosong "".

Line 198: Validasi Input Kosong: if (username.isEmpty() || password.isEmpty() || nickname.isEmpty()) {

- Memeriksa apakah salah satu dari username, password, atau nickname kosong (.isEmpty()). Operator || (OR) berarti kondisi akan true jika salah satu dari ketiganya kosong.

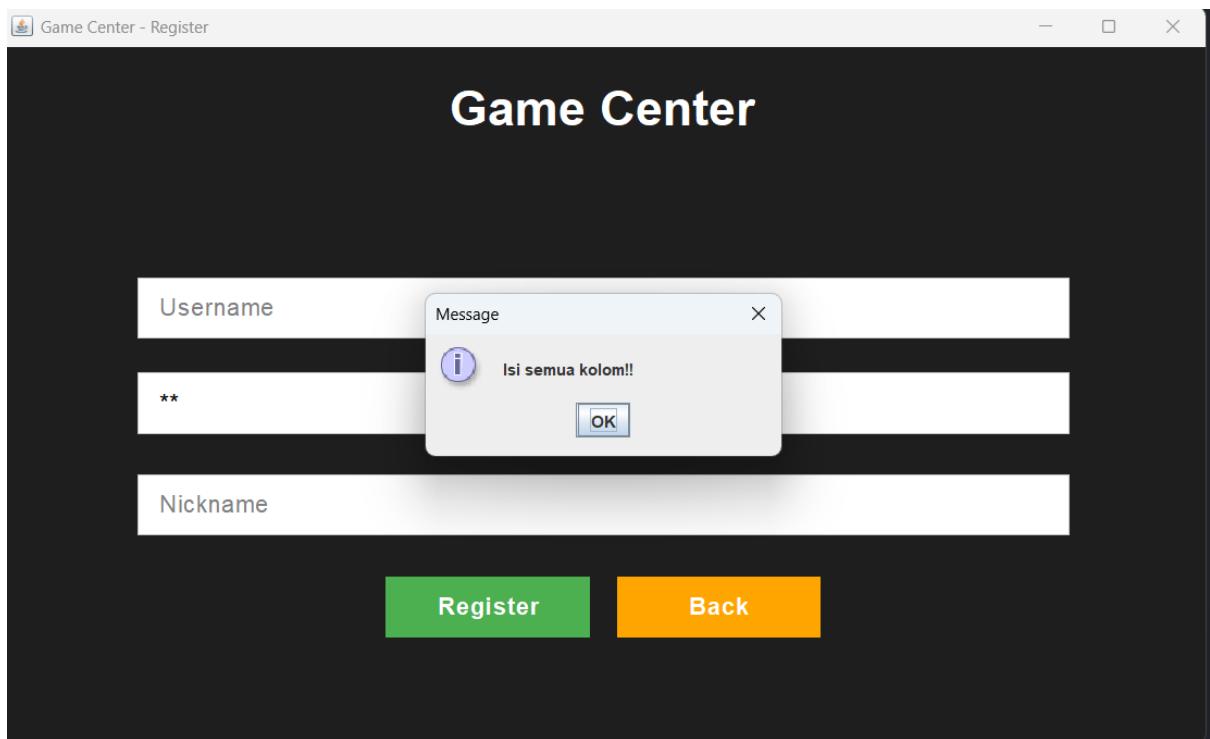
Line 199: Menampilkan Pesan Error: JOptionPane.showMessageDialog(this, message:"Isi semua kolom!");

- Jika ada input yang kosong, sebuah kotak dialog pesan akan ditampilkan kepada pengguna dengan teks "Isi semua kolom!". this menunjukkan bahwa dialog akan berpusat pada komponen saat ini.

Line 200: Menghentikan Eksekusi: return;

- Jika ada input yang kosong, metode akan berhenti di sini dan tidak melanjutkan ke logika pendaftaran.

Output : jika terdapat field registrasi yang tidak diisi akan muncul pop up pesan.



```

203     try (Connection conn = DBConnection.getConnection()) {
204         String checkSql = "SELECT username FROM users WHERE username = ?";
205         PreparedStatement checkStmt = conn.prepareStatement(checkSql);
206         checkStmt.setString(parameterIndex:1, username);
207
208         if (checkStmt.executeQuery().next()) {
209             JOptionPane.showMessageDialog(this, message:"Username sudah tersedia");
210             return;
211         }
212
213         String insertSql = "INSERT INTO users (username, password, nickname) VALUES (?, ?, ?)";
214         PreparedStatement insertStmt = conn.prepareStatement(insertSql, Statement.RETURN_GENERATED_KEYS);
215         insertStmt.setString(parameterIndex:1, username);
216         insertStmt.setString(parameterIndex:2, password);
217         insertStmt.setString(parameterIndex:3, nickname);
218
219         int affectedRows = insertStmt.executeUpdate();
220
221         if (affectedRows > 0) {
222             JOptionPane.showMessageDialog(this, message:"Registrasi Berhasil");
223             new LoginFrame().setVisible(b:true);
224             dispose();
225         }
226     } catch (SQLException ex) {
227         JOptionPane.showMessageDialog(this, "Database error: " + ex.getMessage());
228         ex.printStackTrace();
229     }
230 }
231 }
```

Line 203: Blok try: try (Connection conn = DBConnection.getConnection() {

- Memulai blok try-with-resources. Ini memastikan bahwa Connection (koneksi database) akan ditutup secara otomatis setelah blok try selesai, bahkan jika ada pengecualian. DBConnection.getConnection() digunakan untuk mendapatkan koneksi ke database.

Line 204: Query SQL untuk Mengecek Username: String checkSql = "SELECT username FROM users WHERE username = ?";

- Mendefinisikan string checkSql yang berisi perintah SQL untuk memilih username dari tabel users di mana username cocok dengan nilai yang diberikan (ditandai dengan ? sebagai *placeholder*). Query ini digunakan untuk memeriksa apakah username sudah ada.

Line 205: Mempersiapkan Pernyataan Cek Username: PreparedStatement checkStmt = conn.prepareStatement(checkSql);

- Membuat objek PreparedStatement bernama checkStmt dari objek Connection (conn) dan query checkSql. PreparedStatement lebih aman dari serangan *SQL injection*.

Line 206: Mengatur Parameter Username: checkStmt.setString(parameterIndex:1, username);

- Mengatur nilai untuk *placeholder* pertama (?) di checkSql dengan nilai variabel username yang diambil sebelumnya.

Line 208: Mengeksekusi Query Cek Username: if (checkStmt.executeQuery().next()) {

- Mengeksekusi query checkSql menggunakan checkStmt.executeQuery(). Metode .next() pada ResultSet (hasil dari query) akan mengembalikan true jika ada baris data yang ditemukan (yang berarti username sudah ada), dan false jika tidak.

Line 209: Menampilkan Pesan Username Tersedia: JOptionPane.showMessageDialog(this, message:"Username sudah tersedia");

- Jika username sudah ada, sebuah kotak dialog pesan akan ditampilkan kepada pengguna dengan teks "Username sudah tersedia".

Line 210: Menghentikan Eksekusi: return;

- Jika username sudah ada, metode akan berhenti di sini dan tidak melanjutkan ke proses pendaftaran.

Line 211: Penutup blok if.

Line 213: Query SQL untuk Menyisipkan Pengguna Baru: String insertSql = "INSERT INTO users (username, password, nickname) VALUES (?, ?, ?)";

- Mendefinisikan string insertSql yang berisi perintah SQL untuk menyisipkan data username, password, dan nickname ke dalam tabel users. Tiga tanda tanya (?) adalah *placeholder* untuk nilai yang akan disisipkan.

Line 214: Mempersiapkan Pernyataan Sisipkan Pengguna: PreparedStatement insertStmt = conn.prepareStatement(insertSql, Statement.RETURN_GENERATED_KEYS);

- Membuat objek PreparedStatement bernama insertStmt dari objek Connection dan query insertSql. Statement.RETURN_GENERATED_KEYS ditambahkan untuk mengindikasikan bahwa database harus mengembalikan kunci yang dibuat secara otomatis (misalnya, ID pengguna) setelah operasi penyisipan.

Line 215: Mengatur Parameter Username (untuk Insert):

```
insertStmt.setString(parameterIndex:1, username);
```

- Mengatur nilai untuk *placeholder* pertama (?) di insertSql dengan nilai variabel username.

Line 216: Mengatur Parameter Password (untuk Insert):

```
insertStmt.setString(parameterIndex:2, password);
```

- Mengatur nilai untuk *placeholder* kedua (?) di insertSql dengan nilai variabel password.

Line 217: Mengatur Parameter Nickname (untuk Insert):

```
insertStmt.setString(parameterIndex:3, nickname);
```

- Mengatur nilai untuk *placeholder* ketiga (?) di insertSql dengan nilai variabel nickname.

Line 219: Mengeksekusi Pembaruan: int affectedRows = insertStmt.executeUpdate();

- Mengeksekusi perintah SQL INSERT menggunakan insertStmt.executeUpdate(). Metode ini mengembalikan jumlah baris yang terpengaruh (disisipkan dalam kasus ini).

Line 221: Pemeriksaan Keberhasilan Registrasi: if (affectedRows > 0) {

- Memeriksa apakah affectedRows lebih besar dari 0, yang berarti setidaknya satu baris berhasil disisipkan.

Line 222: Menampilkan Pesan Sukses: JOptionPane.showMessageDialog(this, message:"Registrasi Berhasil");

- Jika pendaftaran berhasil, sebuah kotak dialog pesan akan ditampilkan kepada pengguna dengan teks "Registrasi Berhasil".

Line 223: Mengarahkan ke Halaman Login: new LoginForm().setVisible(b:true);

- Membuat instance baru dari LoginForm (kemungkinan jendela login utama) dan membuatnya terlihat, mengarahkan pengguna ke halaman login setelah pendaftaran.

Line 224: Menutup Jendela Saat Ini: dispose();

- Menutup dan membebaskan sumber daya dari jendela RegisterFrame saat ini.

Line 226: Blok catch: catch (SQLException ex) {

- Menangkap SQLException jika ada kesalahan yang terjadi selama operasi database di blok try.

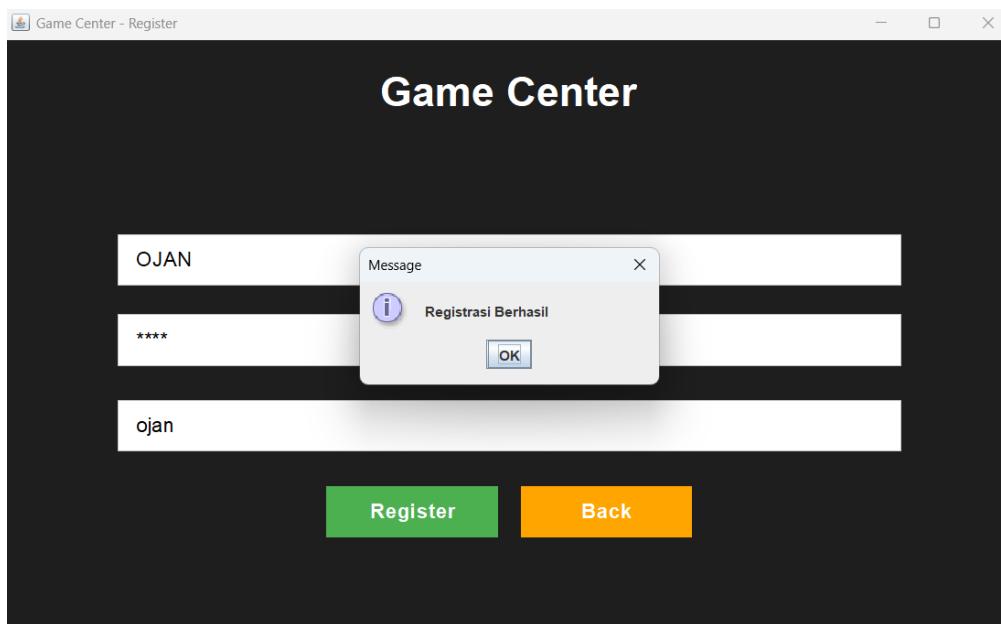
Line 227: Menampilkan Pesan Error Database: JOptionPane.showMessageDialog(this, "Database error: " + ex.getMessage());

- Menampilkan kotak dialog pesan kepada pengguna dengan teks "Database error: " diikuti dengan detail pesan error dari objek SQLException.

Line 228: Mencetak Stack Trace: ex.printStackTrace();

- Mencetak *stack trace* dari pengecualian ke konsol standar error. Ini sangat berguna untuk debugging, karena menunjukkan di mana dan bagaimana pengecualian terjadi.

Output jika registrasi berhasil :



BAB III

PENUTUP

3.1 KESIMPULAN

Pengembangan sistem GameCenter berbasis Java GUI dengan integrasi MySQL telah berhasil menciptakan platform hiburan digital yang komprehensif dan efisien. Sistem ini dirancang menggunakan pendekatan pemrograman berorientasi objek (OOP), yang meliputi penerapan *class*, *object*, *enkapsulasi*, *inheritance*, *polimorfisme*, dan *exception handling*, sehingga menghasilkan struktur kode yang modular, terstruktur, dan mudah untuk dikembangkan lebih lanjut.

Fitur utama yang diimplementasikan mencakup login multi-role untuk admin dan pengguna biasa, pendaftaran akun baru, serta dashboard yang menampilkan skor pribadi dan peringkat global. Pengguna dapat memainkan game seperti Tic Tac Toe (dengan papan 15x15 dan deteksi 5 simbol berurutan untuk kemenangan) dan Snake Game (dengan papan 20x20 dan penyimpanan skor tertinggi). Admin memiliki kemampuan untuk mengelola data pengguna dan skor game secara akurat dan terorganisir. Antarmuka pengguna dirancang modern, responsif, dan intuitif menggunakan Java Swing, memfasilitasi pengalaman bermain yang lancar dan pelacakan skor yang transparan. Integrasi MySQL memastikan penyimpanan data pengguna dan riwayat permainan yang terstruktur dan aman.

Secara keseluruhan, sistem GameCenter ini menjadi bukti nyata penerapan konsep OOP dan rekayasa perangkat lunak dalam memecahkan masalah di bidang hiburan digital, memberikan solusi yang efisien dan fleksibel bagi pengguna dan admin.

3.2 SARAN

Berdasarkan hasil pengembangan dan pengujian sistem GameCenter ini, beberapa saran untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Penambahan Variasi Game: Mengembangkan dan mengintegrasikan lebih banyak jenis game kasual dapat meningkatkan daya tarik dan retensi pengguna. Hal ini bisa mencakup game puzzle, arcade, atau game kartu sederhana.
2. Fitur Sosial dan Komunitas: Menambahkan fitur chat antar pemain, grup diskusi, atau kemampuan untuk menantang teman secara langsung (selain fitur pemain kedua di Tic Tac Toe) dapat memperkuat aspek komunitas dalam GameCenter.
3. Personalisasi dan Kustomisasi: Memberikan opsi personalisasi antarmuka pengguna (misalnya, tema, avatar) atau kustomisasi dalam game (misalnya, skin ular, simbol Tic Tac Toe) dapat meningkatkan pengalaman pengguna.
4. Optimasi Performa Game: Untuk game yang lebih kompleks atau di masa depan, perlu dilakukan optimasi performa agar game berjalan lebih lancar, terutama pada perangkat dengan spesifikasi terbatas.

5. Peningkatan Keamanan: Meskipun sudah ada validasi dasar dan *exception handling*, keamanan sistem dapat ditingkatkan dengan implementasi enkripsi password yang lebih kuat dan penanganan potensi kerentanan SQL injection yang lebih mendalam.
6. Ekspansi ke Platform Lain: Mempertimbangkan pengembangan versi web atau mobile dari GameCenter untuk menjangkau audiens yang lebih luas.
7. Sistem Notifikasi: Menambahkan notifikasi untuk pembaruan papan peringkat, pesan dari teman, atau *event* dalam game dapat menjaga pengguna tetap terlibat.