

LAPORAN TUGAS BESAR UJIAN AKHIR SEMESTER
PEMROGRAMAN BERORIENTASI OBJEK



Disusun Oleh :

Asyifa Rahmina Yudi

2311521007

Dosen Pengampu :

Jeofil Rahmadoni.,M.Kom

DEPARTEMEN SISTEM INFORMASI
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS

2024

A. Kodingan

- **Kelas Hotel.java**

```
import java.util.ArrayList; // Mengimpor ArrayList dari Collection Framework

// Interface Manajemen
interface Manajemen {
    void KelolaKamar(); // Metode untuk mengelola kamar
    void lihatDetail(); // Metode untuk melihat detail
}

// Superclass Hotel yang mengimplementasikan Manajemen
class Hotel implements Manajemen {
    protected String namaHotel; // Nama hotel
    protected String alamat; // Alamat hotel
    protected ArrayList<Kamar> daftarKamar; // Daftar kamar menggunakan ArrayList

    // Constructor
    public Hotel(String namaHotel, String alamat) {
        this.namaHotel = namaHotel;
        this.alamat = alamat;
        this.daftarKamar = new ArrayList<>(); // Inisialisasi ArrayList
    }

    // Method untuk menambahkan kamar ke daftar
    public void tambahKamar(Kamar kamar) {
        daftarKamar.add(kamar);
        System.out.println("Kamar berhasil ditambahkan ke hotel " + namaHotel); // Manipulasi
        // string untuk menggabungkan teks dengan variabel namaHotel
    }

    // Method untuk menampilkan semua kamar di hotel
    public void tampilkanKamar() {
        // Percabangan untuk mengecek apakah daftar kamar kosong
        if (daftarKamar.isEmpty()) {
            // Manipulasi string untuk menyusun pesan ketika tidak ada kamar
            System.out.println("Belum ada kamar yang terdaftar di hotel " + namaHotel);
        } else {
            // Manipulasi string untuk menampilkan judul daftar kamar
            System.out.println("Daftar kamar di hotel " + namaHotel + " :");
            // Perulangan untuk iterasi melalui semua objek Kamar di daftarKamar
            for (Kamar kamar : daftarKamar) {
                kamar.lihatDetail(); // Memanggil metode lihatDetail dari Kamar
                // Manipulasi string untuk menampilkan pemisah antar kamar
                System.out.println("-----");
            }
        }
    }
}
```

```

    }
}

// Implementasi metode dari interface Manajemen
@Override
public void KelolaKamar() {
    // Manipulasi string untuk menyusun pesan terkait manajemen hotel
    System.out.println("Manajemen hotel sedang berjalan untuk " + namaHotel);
}

@Override
public void lihatDetail() {
    // Manipulasi string untuk menampilkan detail hotel
    System.out.println("Detail Hotel:");
    System.out.println("Nama Hotel: " + namaHotel); // Manipulasi string untuk nama hotel
    System.out.println("Alamat: " + alamat); // Manipulasi string untuk alamat hotel
}
}

```

- **Kelas Kamar.java**

```

// Subclass Kamar yang mewarisi kelas Hotel
// Kelas ini menggunakan inheritance dari superclass Hotel dan mengimplementasikan
interface Manajemen

class Kamar extends Hotel implements Manajemen {
    // Variabel instance untuk menyimpan data kamar
    private final int nomorKamar; // Menyimpan nomor kamar
    private final String tipeKamar; // Menyimpan tipe kamar
    private final double hargaPerMalam; // Menyimpan harga per malam
    private boolean tersedia; // Menyimpan status ketersediaan kamar

    // Constructor
    // Constructor untuk menginisialisasi data kamar dan menggunakan constructor superclass
    dengan kata kunci super

    public Kamar(String namaHotel, String alamat, int nomorKamar, String tipeKamar, double
    hargaPerMalam) {
        super(namaHotel, alamat); // Memanggil constructor dari superclass Hotel
        this.nomorKamar = nomorKamar;
    }
}

```

```

        this.tipeKamar = tipeKamar;

        this.hargaPerMalam = hargaPerMalam;

        this.tersedia = true; // Default: kamar tersedia
    }

    // Implementasi metode Kelola Kamar dari interface Manajemen

    @Override
    public void KelolaKamar() {
        // Percabangan untuk mengecek ketersediaan kamar

        if (tersedia) { // Jika tersedia

            tersedia = false; // Ubah status menjadi tidak tersedia

            System.out.println("Kamar " + nomorKamar + " kamar tersedia."); // Manipulasi String
            (penggabungan teks)

        } else { // Jika tidak tersedia

            System.out.println("Kamar " + nomorKamar + " sudah tidak tersedia."); // Manipulasi
            String

        }

    }

    // Implementasi metode lihatDetail dari interface Manajemen

    @Override
    public void lihatDetail() {
        // Menampilkan detail kamar (manipulasi String untuk penggabungan teks)

        System.out.println("Detail Kamar:");

        System.out.println("Nomor Kamar: " + nomorKamar); // Manipulasi String

        System.out.println("Tipe Kamar: " + tipeKamar); // Manipulasi String

        System.out.println("Harga per Malam: Rp " + hargaPerMalam); // Manipulasi String

        System.out.println("Tersedia: " + (tersedia ? "Ya" : "Tidak")); // Percabangan ternary
        untuk status ketersediaan

    }

```

```
}
```

- **Kelas Main.java**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.InputMismatchException;
import java.util.Random;
import java.util.Scanner;

public class Main {
    private static Connection koneksi; // Variabel statik untuk menyimpan koneksi database

    private static final String PASSWORD = "admin123"; // Password yang harus dimasukkan

    // Method untuk membuat koneksi ke database
    @SuppressWarnings("CallToPrintStackTrace")
    public static void koneksiDatabase() {
        try {
            // Memuat driver JDBC untuk MySQL
            Class.forName("com.mysql.cj.jdbc.Driver"); // Constructor: Memuat driver untuk
koneksi
            // String: URL koneksi ke database
            String url =
"jdbc:mysql://localhost:3306/hotel_db?useSSL=false&serverTimezone=UTC";
```

```

String user = "root"; // String: Username database

String password = ""; // String: Password database


// Membuka koneksi ke database

koneksi = DriverManager.getConnection(url, user, password); // Constructor: Membuat
koneksi menggunakan DriverManager

    System.out.println("Koneksi ke database berhasil."); // String: Pesan sukses
} catch (ClassNotFoundException | SQLException e) { // Exception Handling: Menangkap
kesalahan

    System.out.println("Koneksi ke database gagal."); // String: Pesan error
    e.printStackTrace(); // Menampilkan error stack trace untuk debugging
    System.exit(0); // Percabangan: Keluar dari program jika koneksi gagal
}
}


@SuppressWarnings("CallToPrintStackTrace")
public static void tutupKoneksi() {
    try {
        // Percabangan: Memeriksa apakah koneksi tidak null dan belum ditutup
        if (koneksi != null && !koneksi.isClosed()) {
            koneksi.close(); // Menutup koneksi ke database
            System.out.println("Koneksi database ditutup."); // String: Pesan sukses
        }
    } catch (SQLException e) { // Exception Handling: Menangkap kesalahan saat menutup
koneksi

        System.out.println("Gagal menutup koneksi."); // String: Pesan error
        e.printStackTrace(); // Menampilkan error stack trace untuk debugging
    }
}


// Verifikasi password

```

```

public static void verifikasiPassword(Scanner scanner) {
    // Meminta pengguna memasukkan password

    System.out.print("Masukkan password untuk mengakses sistem: "); // String:
Menampilkan pesan ke pengguna

    String inputPassword = scanner.nextLine(); // String: Membaca input password dari
pengguna

    // Percabangan: Memeriksa apakah password yang dimasukkan cocok dengan password
yang disimpan
    if (!inputPassword.equals(PASSWORD)) {
        System.out.println("Password salah. Program akan keluar."); // Pesan jika password
salah

        System.exit(0); // Menghentikan program jika password salah
    }

    // Jika password benar

    System.out.println("Password benar "); // Pesan jika password benar

    System.out.println("Selamat datang di sistem manajemen hotel "); // String:
Menampilkan pesan selamat datang
}

// Fungsi CAPTCHA
public static void captcha(Scanner scanner) {
    // Membuat objek Random untuk menghasilkan angka acak

    Random random = new Random(); // Constructor: Membuat instance Random

    int angka1 = random.nextInt(50) + 1; // Perhitungan Matematika: Menghasilkan angka
acak 1-50

    int angka2 = random.nextInt(50) + 1; // Perhitungan Matematika: Menghasilkan angka
acak 1-50

    int hasil = angka1 + angka2; // Perhitungan Matematika: Menjumlahkan angka1 dan
angka2

```

```

// Menampilkan soal CAPTCHA kepada pengguna

System.out.println("==== Verifikasi CAPTCHA ====");

System.out.println("Berapa hasil dari: " + angka1 + " + " + angka2 + "?"); // String:
Digunakan untuk menyusun teks soal CAPTCHA

System.out.print("Jawaban Anda: ");

try {
    // Membaca jawaban pengguna

    int jawaban = scanner.nextInt(); // Exception Handling: Potensi
InputMismatchException jika input tidak berupa angka

    scanner.nextLine(); // Membersihkan buffer input

    // Percabangan: Memeriksa apakah jawaban pengguna benar atau salah
    if (jawaban != hasil) {

        System.out.println("CAPTCHA salah. Program akan keluar."); // Pesan jika
jawaban salah

        System.exit(0); // Menghentikan program jika CAPTCHA salah
    }

    // Jika jawaban benar

    System.out.println("CAPTCHA benar "); // Pesan jika CAPTCHA benar

    System.out.println("Selamat datang! ");

} catch (InputMismatchException e) { // Exception Handling: Menangkap kesalahan
input yang tidak sesuai

    System.out.println("Input tidak valid. Program akan keluar."); // Pesan jika input tidak
valid

    System.exit(0); // Menghentikan program
}
}

// Fungsi Create: Menambah data kamar ke database

```



```

@SuppressWarnings("CallToPrintStackTrace")

public static void tambahKamar(int nomorKamar, String tipeKamar, double
hargaPerMalam) {

    // Query SQL untuk memeriksa apakah nomor kamar sudah ada
    String checkSql = "SELECT 1 FROM kamar WHERE nomor_kamar = ?";

    try (PreparedStatement checkStmt = koneksi.prepareStatement(checkSql)) {

        // Exception handling: Membuka statement SQL untuk pengecekan
        checkStmt.setInt(1, nomorKamar); // Mengatur parameter query untuk nomor kamar
        ResultSet rs = checkStmt.executeQuery(); // Menjalankan query dan menyimpan
        hasilnya

        // Percabangan: Memeriksa apakah nomor kamar sudah ada di database
        if (rs.next()) {

            System.out.println("Nomor kamar sudah ada, tidak ditambahkan."); // Pesan jika
            nomor kamar sudah ada

            return; // Keluar dari metode tanpa menambahkan data baru
        }

        // Query SQL untuk menambahkan data kamar baru ke tabel
        String sql = "INSERT INTO kamar (nomor_kamar, tipe_kamar, harga_per_malam,
tersedia) VALUES (?, ?, ?, ?)";

        try (PreparedStatement stmt = koneksi.prepareStatement(sql)) {

            // Mengatur parameter untuk query INSERT
            stmt.setInt(1, nomorKamar); // Parameter nomor kamar
            stmt.setString(2, tipeKamar); // Parameter tipe kamar
            stmt.setDouble(3, hargaPerMalam); // Parameter harga per malam
            stmt.setBoolean(4, true); // Parameter tersedia (default: true)

            stmt.executeUpdate(); // Menjalankan query untuk menambahkan data
            System.out.println("Data kamar berhasil ditambahkan."); // Pesan sukses
        } catch (SQLException e) { // Exception handling untuk query INSERT

```

```

        System.out.println("Gagal menambahkan data kamar."); // Pesan error
        e.printStackTrace(); // Menampilkan rincian error untuk debugging
    }
} catch (SQLException e) { // Exception handling untuk pengecekan data
    System.out.println("Gagal memeriksa data kamar."); // Pesan error
    e.printStackTrace(); // Menampilkan rincian error untuk debugging
}
}

// Fungsi Read: Membaca data kamar dari database
@SuppressWarnings("CallToPrintStackTrace")
public static void bacaKamar() {
    // Query SQL untuk membaca semua data dari tabel 'kamar'
    String sql = "SELECT nomor_kamar, tipe_kamar, harga_per_malam, tersedia FROM kamar";

    try (Statement stmt = koneksi.createStatement(); ResultSet rs = stmt.executeQuery(sql))
    {
        // Exception handling: Membuka statement dan eksekusi query
        // Perulangan: Mengiterasi hasil query menggunakan while loop
        while (rs.next()) {
            // Menampilkan data kamar satu per satu
            System.out.println("Nomor Kamar: " + rs.getInt("nomor_kamar")); // Menampilkan nomor kamar

            System.out.println("Tipe Kamar: " + rs.getString("tipe_kamar")); // Menampilkan tipe kamar

            System.out.println("Harga per Malam: Rp " + rs.getDouble("harga_per_malam"));
            // Menampilkan harga per malam

            // Percabangan: Menentukan apakah kamar tersedia atau tidak
            System.out.println("Tersedia: " + (rs.getBoolean("tersedia") ? "Ya" : "Tidak"));
        }
    }
}

```

```

        System.out.println("-----");
    }
} catch (SQLException e) { // Exception handling: Menangkap kesalahan saat eksekusi
query SQL
    System.out.println("Gagal membaca data kamar."); // Menampilkan pesan error
    e.printStackTrace(); // Menampilkan rincian error untuk debugging
}
}

```

```

// Fungsi Update: Memperbarui data kamar di database
@SuppressWarnings("CallToPrintStackTrace")
public static void updateKamar(Scanner scanner) {
    // Menampilkan daftar kamar yang tersedia
    bacaKamar();

    // Meminta pengguna memasukkan nomor kamar yang akan diperbarui
    System.out.print("Masukkan nomor kamar yang ingin diperbarui: ");
    int nomorKamar = scanner.nextInt();
    scanner.nextLine(); // Membersihkan buffer input

    // Percabangan: Memeriksa apakah nomor kamar valid
    if (!cekNomorKamar(nomorKamar)) {
        System.out.println("Nomor kamar tidak ditemukan."); // Pesan jika nomor kamar
tidak ditemukan
        return; // Keluar dari metode jika nomor kamar tidak valid
    }

    // Meminta data baru untuk memperbarui kamar
    System.out.print("Masukkan tipe kamar baru: ");
    String tipeBaru = scanner.nextLine(); // Membaca tipe kamar baru

```

```

System.out.print("Masukkan harga baru per malam: ");

double hargaBaru = scanner.nextDouble(); // Membaca harga kamar baru
scanner.nextLine(); // Membersihkan buffer input

// Query SQL untuk memperbarui data kamar

String sql = "UPDATE kamar SET tipe_kamar = ?, harga_per_malam = ? WHERE
nomor_kamar = ?";

try (PreparedStatement stmt = koneksi.prepareStatement(sql)) { // Exception handling:
Membuka koneksi SQL

    // Mengatur parameter query SQL

    stmt.setString(1, tipeBaru); // Parameter tipe kamar

    stmt.setDouble(2, hargaBaru); // Parameter harga kamar

    stmt.setInt(3, nomorKamar); // Parameter nomor kamar

    // Menjalankan query update

    stmt.executeUpdate();

    System.out.println("Data kamar berhasil diperbarui."); // Pesan sukses
} catch (SQLException e) { // Exception handling: Menangkap kesalahan SQL

    System.out.println("Gagal memperbarui data kamar."); // Pesan jika terjadi error

    e.printStackTrace(); // Menampilkan detail error untuk debugging

}

}

// Fungsi Delete: Menghapus data kamar dari database
@SuppressWarnings("CallToPrintStackTrace")
public static void hapusKamar(int nomorHapus) {

    String sql = "DELETE FROM kamar WHERE nomor_kamar = ?";

    //Menggunakan try-with-resources untuk memastikan statement ditutup secara otomatis
    setelah selesai.

    try (PreparedStatement stmt = koneksi.prepareStatement(sql)) {

```

```

        stmt.setInt(1, nomorHapus);

        stmt.executeUpdate();

        System.out.println("Data kamar dengan nomor " + nomorHapus + " berhasil
dihapus.");
    } catch (SQLException e) { // Menangkap exception jika ada kesalahan dalam eksekusi
SQL
        System.out.println("Gagal menghapus data kamar.");

        e.printStackTrace();//Menampilkan rincian kesalahan (stack trace) untuk membantu
debugging jika terjadi masalah saat eksekusi query
    }
}

// Fungsi untuk mengecek apakah nomor kamar ada di database
@SuppressWarnings("CallToPrintStackTrace")
public static boolean cekNomorKamar(int nomorKamar) {
    String sql = "SELECT 1 FROM kamar WHERE nomor_kamar = ?";

    //try-catch digunakan untuk menangani pengecualian yang mungkin terjadi selama
eksekusi query SQL
    try (PreparedStatement stmt = koneksi.prepareStatement(sql)) {
        stmt.setInt(1, nomorKamar);

        try (ResultSet rs = stmt.executeQuery()) {
            return rs.next(); // Mengembalikan true jika nomor kamar ditemukan
        }
    } catch (SQLException e) { //Exception Handling
        System.out.println("Gagal mengecek nomor kamar.");
        e.printStackTrace();
    }

    return false; // Mengembalikan false jika terjadi kesalahan
}

// Fungsi Tanggal dengan validasi nomor kamar
public static void tanggal(Scanner scanner) {

```

```

try {
    System.out.print("Masukkan nomor kamar: ");
    int nomorKamar = scanner.nextInt();
    scanner.nextLine(); // Bersihkan buffer input

    if (!cekNomorKamar(nomorKamar)) { //Percabangan (if statement)
        System.out.println("Nomor kamar tidak ditemukan. Silakan masukkan nomor yang
valid.");
        return;
    }

    System.out.print("Masukkan tanggal check-in (dd-MM-yyyy): ");
    String tanggalCheckIn = scanner.nextLine(); // Gunakan nextLine untuk membaca
input
    System.out.print("Masukkan jumlah malam menginap: ");
    int jumlahMalam = scanner.nextInt();
    scanner.nextLine(); // Bersihkan buffer input setelah membaca angka

    SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");

    Date date = format.parse(tanggalCheckIn); // Mengubah string tanggal menjadi objek
Date

    Calendar calendar = Calendar.getInstance(); // Membuat objek Calendar yang dapat
digunakan untuk manipulasi tanggal
    calendar.setTime(date); // Menetapkan tanggal check-in ke objek calendar
    calendar.add(Calendar.DAY_OF_MONTH, jumlahMalam); // perhitungan
matematis terkait menambahkan jumlah malam yang diinputkan oleh pengguna ke tanggal
check-in

    Date tanggalCheckOut = calendar.getTime(); // Mendapatkan tanggal check-out
sebagai objek Date

    System.out.println("Nomor kamar: " + nomorKamar);
    System.out.println("Tanggal check-in: " + format.format(date));

```

```

        System.out.println("Tanggal check-out: " + format.format(tanggalCheckOut));

    } catch (ParseException e) {
        // Menangani kesalahan jika format tanggal tidak valid
        System.out.println("Format tanggal tidak valid. Gunakan format dd-MM-yyyy.");
    } catch (InputMismatchException e) {
        // Menangani kesalahan jika input bukan angka yang diharapkan
        System.out.println("Input tidak valid.");
        scanner.nextLine(); // Bersihkan buffer input jika terjadi kesalahan
    }
}

public static void main(String[] args) {
    try (Scanner scanner = new Scanner(System.in)) {
        koneksiDatabase();
        verifikasiPassword(scanner);
        captcha(scanner);

        int pilihan = -1;
        // Perulangan untuk menampilkan menu dan menangani input pilihan pengguna
        do {
            // Tampilkan menu hanya sekali per iterasi
            System.out.println("\n=== Manajemen Kamar Hotel ===");
            System.out.println("1. Tambah Kamar");
            System.out.println("2. Lihat Data Kamar");
            System.out.println("3. Perbarui Data Kamar");
            System.out.println("4. Hapus Data Kamar");
            System.out.println("5. Tanggal Check-in dan Check-out");
            System.out.println("6. Keluar");

```

```
System.out.print("Pilih menu: ");

try {
    pilihan = scanner.nextInt();
    scanner.nextLine(); // Bersihkan buffer setelah membaca angka

    switch (pilihan) {
        case 1 -> {
            System.out.print("Masukkan nomor kamar: ");
            int nomorKamar = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Masukkan tipe kamar: ");
            String tipeKamar = scanner.nextLine();
            System.out.print("Masukkan harga per malam: ");
            double hargaPerMalam = scanner.nextDouble();
            scanner.nextLine();
            tambahKamar(nomorKamar, tipeKamar, hargaPerMalam);
        }
        case 2 -> bacaKamar();
        case 3 -> updateKamar(scanner);
        case 4 -> {
            System.out.print("Masukkan nomor kamar yang ingin dihapus: ");
            int nomorHapus = scanner.nextInt();
            scanner.nextLine();
            hapusKamar(nomorHapus);
        }
        case 5 -> tanggal(scanner);
        case 6 -> System.out.println("Terima kasih telah menggunakan sistem
management hotel.");
        default -> System.out.println("Pilihan tidak valid. Silakan coba lagi.");
    }
}
```



```

    } catch (InputMismatchException e) {
        // Exception handling untuk input yang tidak valid
        System.out.println("Input tidak valid. Masukkan angka.");
        scanner.nextLine(); // Bersihkan buffer untuk mencegah loop terus berjalan
        pilihan = -1; // Mengatur ulang pilihan agar tidak keluar dari loop
    }
} while (pilihan != 6); // Loop akan terus berjalan sampai pengguna memilih untuk
keluar

    tutupKoneksi();
}
}
}

```

B. Penjelasan Program

Program ini merupakan sistem manajemen kamar hotel berbasis Java yang terintegrasi dengan database MySQL. Program diawali dengan menghubungkan sistem ke database melalui driver JDBC. Jika koneksi berhasil, pesan sukses akan ditampilkan, sedangkan kegagalan akan menyebabkan program keluar. Sistem ini memiliki lapisan keamanan berupa verifikasi password untuk mengakses aplikasi dan CAPTCHA sederhana untuk memvalidasi pengguna. Setelah proses verifikasi, pengguna diarahkan ke menu utama dengan beberapa opsi, yaitu menambah data kamar, melihat daftar kamar, memperbarui data kamar, menghapus data kamar, dan mengelola tanggal check-in serta check-out.

Pada fitur penambahan kamar, pengguna diminta memasukkan informasi seperti nomor kamar, tipe kamar, dan harga per malam, yang kemudian disimpan ke dalam database. Untuk melihat data kamar, sistem akan membaca dan menampilkan informasi lengkap dari database, termasuk nomor kamar, tipe, harga, dan status ketersediaan. Fitur pembaruan memungkinkan pengguna memperbarui tipe dan harga kamar berdasarkan nomor kamar tertentu, sementara fitur penghapusan memungkinkan pengguna menghapus data kamar dengan nomor tertentu. Fitur tanggal check-in dan check-out menghitung tanggal keluar berdasarkan jumlah malam menginap yang dimasukkan pengguna.

Program ini juga memiliki validasi input untuk memastikan data yang dimasukkan sesuai format, seperti nomor kamar yang valid dan format tanggal yang benar. Berbagai kesalahan seperti koneksi database yang gagal, input tidak valid, atau data tidak ditemukan akan ditangani dengan menampilkan pesan kesalahan. Setelah pengguna keluar dari menu, koneksi ke database ditutup untuk memastikan semua sumber daya dilepaskan dengan benar.

➤ Class:

- **Hotel.java:** Sebagai superclass yang berisi atribut dan metode dasar seperti nama hotel dan alamat.
- **Kamar.java:** Subclass dari Hotel yang juga mengimplementasikan interface Manajemen
- **Main.java:** Kelas utama yang mengatur interaksi dengan pengguna, termasuk CRUD dan pengelolaan koneksi database.

➤ **Objec :**

- **Objek Scanner**

Objek scanner digunakan untuk membaca input dari pengguna melalui konsol. Dibuat di dalam try resource block di metode main.

```
Scanner scanner = new Scanner(System.in);
```

- **Objek Connection**

Objek koneksi adalah instance dari kelas Connection dari pustaka JDBC. Objek ini dibuat melalui metode DriverManager.getConnection() di dalam metode koneksiDatabase.

```
Connection koneksi;
```

- **Objek Random**

Digunakan untuk menghasilkan angka acak untuk verifikasi CAPTCHA di metode captcha

```
Random random = new Random();
```

- **Objek PreparedStatement dan Statement**

Objek PreparedStatement dibuat melalui metode koneksi.prepareStatement() saat melakukan operasi CRUD di database.

```
PreparedStatement stmt = koneksi.prepareStatement(sql);
```

Objek Statement digunakan untuk membaca data dari database

```
Statement stmt = koneksi.createStatement();
```

- **Objek ResultSet**

Objek ResultSet digunakan untuk menyimpan hasil query dari database

```
ResultSet rs = stmt.executeQuery();
```

- **Objek SimpleDateFormat**

Objek format adalah instance dari kelas SimpleDateFormat

```
SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
```

- **Objek Calendar**

Objek calendar adalah instance dari kelas Calendar. Digunakan untuk memanipulasi tanggal, seperti menambahkan jumlah malam menginap di metode tanggal.

```
Calendar calendar = Calendar.getInstance();
```

- **Objek Date**

Objek date adalah instance dari kelas Date. Digunakan untuk menyimpan tanggal check-in dan check-out di metode tanggal.

```
Date date = format.parse(tanggalCheckIn);
Date tanggalCheckOut = calendar.getTime();
```

- **Constructor**

- **Constructor pada kelas Hotel**

Fungsi: Menginisialisasi atribut namaHotel dan alamat ketika objek dari kelas Hotel dibuat.

```
// Constructor
public Hotel(String namaHotel, String alamat) {
    this.namaHotel = namaHotel;
    this.alamat = alamat;
    this.daftarKamar = new ArrayList<>(); // Inisialisasi ArrayList
}
```

- **Constructor pada Kelas Kamar**

```
public Kamar(String namaHotel, String alamat, int nomorKamar, String
tipeKamar, double hargaPerMalam) {
    super(namaHotel, alamat); // Memanggil constructor dari superclass Hotel
    this.nomorKamar = nomorKamar;
    this.tipeKamar = tipeKamar;
    this.hargaPerMalam = hargaPerMalam;
    this.tersedia = true; // Default: kamar tersedia
}
```

- **Constructor Implisit pada kelas Main**

Random random = new Random();

Constructor: Random(). Digunakan untuk membuat instance dari kelas Random untuk menghasilkan angka acak.

Scanner scanner = new Scanner(System.in);

Constructor: Scanner(InputStream source). Membuat instance dari kelas Scanner untuk membaca input dari konsol (System.in).

Calendar calendar = Calendar.getInstance();

Constructor: Calendar(). digunakan untuk Memanggil factory method getInstance() untuk mendapatkan instance dari subclass Calendar, biasanya GregorianCalendar.

SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");

Constructor: SimpleDateFormat(String pattern). Membuat instance dari kelas SimpleDateFormat untuk memformat tanggal sesuai pola yang diberikan.

- **Constructor Explicit dalam Objek JDBC pada kelas Main**

Dalam koneksi database, ada constructor yang digunakan melalui class JDBC

Connection koneksi = DriverManager.getConnection(url, user, password);

Constructor: DriverManager.getConnection(String url, String user, String password).Membuat koneksi ke database menggunakan URL, username, dan password.

PreparedStatement stmt = koneksi.prepareStatement(sql);

Constructor: Connection.prepareStatement(String sql).Membuat instance dari PreparedStatement untuk eksekusi query SQL.

Statement stmt = koneksi.createStatement();

Constructor: Connection.createStatement().Membuat instance dari Statement untuk menjalankan query SQL.

ResultSet rs = stmt.executeQuery(sql);

Constructor: Tidak langsung terlihat, tapi instance ResultSet dibuat melalui method executeQuery().

➤ Interface dan Implementasi dari Interface

```
// Interface Manajemen
interface Manajemen {
    void KelolaKamar(); //
    void lihatDetail(); // Metode untuk melihat detail kamar
}
```

```
// Implementasi metode kelola Kamar dari interface Manajemen
@Override
public void KelolaKamar() {
    // Percabangan untuk mengecek ketersediaan kamar
    if (tersedia) { // Jika tersedia
        tersedia = false; // Ubah status menjadi tidak tersedia
        System.out.println("Kamar " + nomorKamar + " kamar tersedia. "); //
        Manipulasi String (penggabungan teks)
    } else { // Jika tidak tersedia
        System.out.println("Kamar " + nomorKamar + " sudah tidak tersedia."); //
        Manipulasi String
    }
}
```

```
// Implementasi metode lihatDetail dari interface Manajemen
@Override
public void lihatDetail() {
    // Menampilkan detail kamar (manipulasi String untuk penggabungan teks)
    System.out.println("Detail Kamar:");
    System.out.println("Nomor Kamar: " + nomorKamar); // Manipulasi String
}
```

```

        System.out.println("Tipe Kamar: " + tipeKamar); // Manipulasi String
        System.out.println("Harga per Malam: Rp " + hargaPerMalam); //
Manipulasi String
        System.out.println("Tersedia: " + (tersedia ? "Ya" : "Tidak")); // Percabangan
ternary untuk status ketersediaan
    }
}

```

➤ Inheritance (superclass dan sub class)

• Superclass

```

import java.util.ArrayList;

// Interface Manajemen
interface Manajemen {
    void KelolaKamar(); // Metode untuk mengelola kamar
    void lihatDetail(); // Metode untuk melihat detail
}

// Superclass Hotel yang mengimplementasikan Manajemen
class Hotel implements Manajemen {
    protected String namaHotel; // Nama hotel
    protected String alamat; // Alamat hotel
    protected ArrayList<Kamar> daftarKamar; // Daftar kamar menggunakan
ArrayList

    // Constructor
    public Hotel(String namaHotel, String alamat) {
        this.namaHotel = namaHotel;
        this.alamat = alamat;
        this.daftarKamar = new ArrayList<>(); // Inisialisasi ArrayList
    }

    // Method untuk menambahkan kamar ke daftar
    public void tambahKamar(Kamar kamar) {
        daftarKamar.add(kamar);
        System.out.println("Kamar berhasil ditambahkan ke hotel " + namaHotel);
    }

    // Method untuk menampilkan semua kamar di hotel
    public void tampilkanKamar() {
        if (daftarKamar.isEmpty()) {
            System.out.println("Belum ada kamar yang terdaftar di hotel " +
namaHotel);
        } else {

```

```

        System.out.println("Daftar kamar di hotel " + namaHotel + ":");
        for (Kamar kamar : daftarKamar) {
            kamar.lihatDetail(); // Memanggil metode lihatDetail dari Kamar
            System.out.println("-----");
        }
    }
}

// Implementasi metode dari interface Manajemen
@Override
public void KelolaKamar() {
    System.out.println("Manajemen hotel sedang berjalan untuk " + namaHotel);
}

@Override
public void lihatDetail() {
    System.out.println("Detail Hotel:");
    System.out.println("Nama Hotel: " + namaHotel);
    System.out.println("Alamat: " + alamat);
}
}

```

- **Subclass**

```

// Subclass Kamar yang mewarisi kelas Hotel
// Kelas ini menggunakan inheritance dari superclass Hotel dan
mengimplementasikan interface Manajemen
class Kamar extends Hotel implements Manajemen {
    // Variabel instance untuk menyimpan data kamar
    private final int nomorKamar; // Menyimpan nomor kamar
    private final String tipeKamar; // Menyimpan tipe kamar
    private final double hargaPerMalam; // Menyimpan harga per malam
    private boolean tersedia; // Menyimpan status ketersediaan kamar

    // Constructor
    // Constructor untuk menginisialisasi data kamar dan menggunakan constructor
    superclass dengan kata kunci super
    public Kamar(String namaHotel, String alamat, int nomorKamar, String
    tipeKamar, double hargaPerMalam) {
        super(namaHotel, alamat); // Memanggil constructor dari superclass Hotel
        this.nomorKamar = nomorKamar;
        this.tipeKamar = tipeKamar;
        this.hargaPerMalam = hargaPerMalam;
        this.tersedia = true; // Default: kamar tersedia
    }
}

```

```

// Implementasi metode Kelola Kamar dari interface Manajemen
@Override
public void KelolaKamar() {
    // Percabangan untuk mengecek ketersediaan kamar
    if (tersedia) { // Jika tersedia
        tersedia = false; // Ubah status menjadi tidak tersedia
        System.out.println("Kamar " + nomorKamar + " kamar tersedia. "); //
Manipulasi String (penggabungan teks)
    } else { // Jika tidak tersedia
        System.out.println("Kamar " + nomorKamar + " sudah tidak tersedia."); //
Manipulasi String
    }
}

// Implementasi metode lihatDetail dari interface Manajemen
@Override
public void lihatDetail() {
    // Menampilkan detail kamar (manipulasi String untuk penggabungan teks)
    System.out.println("Detail Kamar:");
    System.out.println("Nomor Kamar: " + nomorKamar); // Manipulasi String
    System.out.println("Tipe Kamar: " + tipeKamar); // Manipulasi String
    System.out.println("Harga per Malam: Rp " + hargaPerMalam); //
Manipulasi String
    System.out.println("Tersedia: " + (tersedia ? "Ya" : "Tidak")); // Percabangan
ternary untuk status ketersediaan
}
}

```

➤ Perulangan dan Percabangan

• Percabangan

```

// Percabangan untuk mengecek ketersediaan kamar
if (tersedia) { // Jika tersedia
    tersedia = false; // Ubah status menjadi tidak tersedia
    System.out.println("Kamar " + nomorKamar + " kamar tersedia."); //
Manipulasi String (penggabungan teks)
} else { // Jika tidak tersedia
    System.out.println("Kamar " + nomorKamar + " sudah tidak tersedia."); //
Manipulasi String
}
}

```

```

if (!inputPassword.equals(PASSWORD)) {
    System.out.println("Password salah. Program akan keluar."); // Pesan jika
password salah
}

```

```
        System.exit(0); // Menghentikan program jika password salah
    }
```

```
// Percabangan: Memeriksa apakah koneksi tidak null dan belum ditutup
if (koneksi != null && !koneksi.isClosed()) {
    koneksi.close(); // Menutup koneksi ke database
    System.out.println("Koneksi database ditutup."); // String: Pesan sukses
}
```

```
// Percabangan: Memeriksa apakah jawaban pengguna benar atau salah
if (jawaban != hasil) {
    System.out.println("CAPTCHA salah. Program akan keluar."); // Pesan
    jika jawaban salah
    System.exit(0); // Menghentikan program jika CAPTCHA salah
}
```

```
// Percabangan: Memeriksa apakah nomor kamar sudah ada di database
if (rs.next()) {
    System.out.println("Nomor kamar sudah ada, tidak ditambahkan."); //
    Pesan jika nomor kamar sudah ada
    return; // Keluar dari metode tanpa menambahkan data baru
}
```

```
// Percabangan: Memeriksa apakah nomor kamar valid
if (!cekNomorKamar(nomorKamar)) {
    System.out.println("Nomor kamar tidak ditemukan."); // Pesan jika nomor
    kamar tidak ditemukan
    return; // Keluar dari metode jika nomor kamar tidak valid
}
```

```
        if (!cekNomorKamar(nomorKamar)) { //Percabangan (if statement)
            System.out.println("Nomor kamar tidak ditemukan. Silakan masukkan
            nomor yang valid.");
            return;
        }
```

- **Perulangan**

```
// Perulangan: Mengiterasi hasil query menggunakan while loop
while (rs.next()) {
    // Menampilkan data kamar satu per satu
    System.out.println("Nomor Kamar: " + rs.getInt("nomor_kamar")); //
    Menampilkan nomor kamar
}
```



```

        System.out.println("Tipe Kamar: " + rs.getString("tipe_kamar")); //
Menampilkan tipe kamar
        System.out.println("Harga per Malam: Rp " +
rs.getDouble("harga_per_malam")); // Menampilkan harga per malam

        // Percabangan: Menentukan apakah kamar tersedia atau tidak
        System.out.println("Tersedia: " + (rs.getBoolean("tersedia") ? "Ya" :
"Tidak"));
        System.out.println("-----");
    }

```

```

// Perulangan untuk menampilkan menu dan menangani input pilihan pengguna
do {
    // Tampilkan menu hanya sekali per iterasi
    System.out.println("\n=== Manajemen Kamar Hotel ===");
    System.out.println("1. Tambah Kamar");
    System.out.println("2. Lihat Data Kamar");
    System.out.println("3. Perbarui Data Kamar");
    System.out.println("4. Hapus Data Kamar");
    System.out.println("5. Tanggal Check-in dan Check-out");
    System.out.println("6. Keluar");
    System.out.print("Pilih menu: ");
}

```

➤ **Perhitungan matematika**

```

int angka1 = random.nextInt(50) + 1; // Perhitungan Matematika: Menghasilkan
angka acak 1-50
    int angka2 = random.nextInt(50) + 1; // Perhitungan Matematika:
Menghasilkan angka acak 1-50
    int hasil = angka1 + angka2; // Perhitungan Matematika: Menjumlahkan
angka1 dan angka2

```

```

calendar.add(Calendar.DAY_OF_MONTH, jumlahMalam); // perhitungan
matematis terkait menambahkan jumlah malam yang diinputkan oleh pengguna
ke tanggal check-in

```

➤ **Manipulasi String and Date**

• **String**

```

// Contoh manipulasi String dengan menggabungkan teks menggunakan operator
+
public void tampilkanInfoHotel() {
    System.out.println("Hotel: " + namaHotel); // Menampilkan nama hotel
    System.out.println("Alamat: " + alamat); // Menampilkan alamat hotel
}
}

```

```

        System.out.println("Kamar " + nomorKamar + " kamar tersedia."); // Manipulasi
String (penggabungan teks)
    } else { // Jika tidak tersedia
        System.out.println("Kamar " + nomorKamar + " sudah tidak tersedia."); //
Manipulasi String
    }
}

```

```

// Menampilkan detail kamar (manipulasi String untuk penggabungan teks)
System.out.println("Detail Kamar:");
System.out.println("Nomor Kamar: " + nomorKamar); // Manipulasi String
System.out.println("Tipe Kamar: " + tipeKamar); // Manipulasi String
System.out.println("Harga per Malam: Rp " + hargaPerMalam); //
Manipulasi String

```

```

System.out.println("Berapa hasil dari: " + angka1 + " + " + angka2 + "?"); //
String: Digunakan untuk menyusun teks soal CAPTCHA

```

- **Date**

```

SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
Date date = format.parse(tanggalCheckIn);

```

```

Calendar calendar = Calendar.getInstance();// Membuat objek Calendar yang
dapat digunakan untuk manipulasi tanggal
    calendar.setTime(date);// Menetapkan tanggal check-in ke objek calendar
    calendar.add(Calendar.DAY_OF_MONTH, jumlahMalam)

```

```

Date tanggalCheckOut = calendar.getTime();// Mendapatkan tanggal check-out
sebagai objek Date
    System.out.println("Nomor kamar: " + nomorKamar);
    System.out.println("Tanggal check-in: " + format.format(date));
    System.out.println("Tanggal check-out: " +
format.format(tanggalCheckOut));

```

- **Exception handling**

```

try { //Exception Handling
    // Memuat driver JDBC untuk MySQL
    Class.forName("com.mysql.cj.jdbc.Driver"); // Constructor: Memuat driver
untuk koneksi
    // String: URL koneksi ke database

```

```

String url =
"jdbc:mysql://localhost:3306/hotel_db?useSSL=false&serverTimezone=UTC";
String user = "root"; // String: Username database
String password = ""; // String: Password database

// Membuka koneksi ke database
koneksi = DriverManager.getConnection(url, user, password); // Constructor:
Membuat koneksi menggunakan DriverManager
System.out.println("Koneksi ke database berhasil."); // String: Pesan sukses
} catch (ClassNotFoundException | SQLException e) { // Exception Handling:
Menangkap kesalahan
System.out.println("Koneksi ke database gagal."); // String: Pesan error
e.printStackTrace(); // Menampilkan error stack trace untuk debugging
System.exit(0); // Percabangan: Keluar dari program jika koneksi gagal
}
}
}

```

```

try { //Exception Handling
// Percabangan: Memeriksa apakah koneksi tidak null dan belum ditutup
if (koneksi != null && !koneksi.isClosed()) {
koneksi.close(); // Menutup koneksi ke database
System.out.println("Koneksi database ditutup."); // String: Pesan sukses
}
} catch (SQLException e) { // Exception Handling: Menangkap kesalahan saat
menutup koneksi
System.out.println("Gagal menutup koneksi."); // String: Pesan error
e.printStackTrace(); // Menampilkan error stack trace untuk debugging
}
}
}

```

```

try {
// Membaca jawaban pengguna
int jawaban = scanner.nextInt(); // Exception Handling: Potensi
InputMismatchException jika input tidak berupa angka
scanner.nextLine(); // Membersihkan buffer input

// Percabangan: Memeriksa apakah jawaban pengguna benar atau salah
if (jawaban != hasil) {
System.out.println("CAPTCHA salah. Program akan keluar."); // Pesan
jika jawaban salah
System.exit(0); // Menghentikan program jika CAPTCHA salah
}

// Jika jawaban benar
System.out.println("CAPTCHA benar "); // Pesan jika CAPTCHA benar
System.out.println("Selamat datang! ");
}

```

```

    } catch (InputMismatchException e) { // Exception Handling: Menangkap
kesalahan input yang tidak sesuai
        System.out.println("Input tidak valid. Program akan keluar."); // Pesan jika
input tidak valid
        System.exit(0); // Menghentikan program
    }
}

```

```

try (PreparedStatement checkStmt = koneksi.prepareStatement(checkSql)) {
    // Exception handling: Membuka statement SQL untuk pengecekan
    checkStmt.setInt(1, nomorKamar); // Mengatur parameter query untuk
nomor kamar
    ResultSet rs = checkStmt.executeQuery(); // Menjalankan query dan
menyimpan hasilnya

    // Percabangan: Memeriksa apakah nomor kamar sudah ada di database
    if (rs.next()) {
        System.out.println("Nomor kamar sudah ada, tidak ditambahkan."); //
Pesan jika nomor kamar sudah ada
        return; // Keluar dari metode tanpa menambahkan data baru
    }

    // Query SQL untuk menambahkan data kamar baru ke tabel
    String sql = "INSERT INTO kamar (nomor_kamar, tipe_kamar,
harga_per_malam, tersedia) VALUES (?, ?, ?, ?)";
    try (PreparedStatement stmt = koneksi.prepareStatement(sql)) {
        // Mengatur parameter untuk query INSERT
        stmt.setInt(1, nomorKamar); // Parameter nomor kamar
        stmt.setString(2, tipeKamar); // Parameter tipe kamar
        stmt.setDouble(3, hargaPerMalam); // Parameter harga per malam
        stmt.setBoolean(4, true); // Parameter tersedia (default: true)

        stmt.executeUpdate(); // Menjalankan query untuk menambahkan data
        System.out.println("Data kamar berhasil ditambahkan."); // Pesan
sukses
    } catch (SQLException e) { // Exception handling untuk query INSERT
        System.out.println("Gagal menambahkan data kamar."); // Pesan error
        e.printStackTrace(); // Menampilkan rincian error untuk debugging
    }
    } catch (SQLException e) { // Exception handling untuk pengecekan data
        System.out.println("Gagal memeriksa data kamar."); // Pesan error
        e.printStackTrace(); // Menampilkan rincian error untuk debugging
    }
}
}

```

```

try (Statement stmt = koneksi.createStatement(); ResultSet rs =
stmt.executeQuery(sql)) {
    // Exception handling: Membuka statement dan eksekusi query
    // Perulangan: Mengiterasi hasil query menggunakan while loop
    while (rs.next()) {
        // Menampilkan data kamar satu per satu
        System.out.println("Nomor Kamar: " + rs.getInt("nomor_kamar")); //
Menampilkan nomor kamar
        System.out.println("Tipe Kamar: " + rs.getString("tipe_kamar")); //
Menampilkan tipe kamar
        System.out.println("Harga per Malam: Rp " +
rs.getDouble("harga_per_malam")); // Menampilkan harga per malam

        // Percabangan: Menentukan apakah kamar tersedia atau tidak
        System.out.println("Tersedia: " + (rs.getBoolean("tersedia") ? "Ya" :
"Tidak"));
        System.out.println("-----");
    }
} catch (SQLException e) { // Exception handling: Menangkap kesalahan
saat eksekusi query SQL
    System.out.println("Gagal membaca data kamar."); // Menampilkan pesan
error
    e.printStackTrace(); // Menampilkan rincian error untuk debugging
}
}

```

```

try (PreparedStatement stmt = koneksi.prepareStatement(sql)) { // Exception
handling: Membuka koneksi SQL
    // Mengatur parameter query SQL
    stmt.setString(1, tipeBaru); // Parameter tipe kamar
    stmt.setDouble(2, hargaBaru); // Parameter harga kamar
    stmt.setInt(3, nomorKamar); // Parameter nomor kamar

    // Menjalankan query update
    stmt.executeUpdate();
    System.out.println("Data kamar berhasil diperbarui."); // Pesan sukses
} catch (SQLException e) { // Exception handling: Menangkap kesalahan
SQL
    System.out.println("Gagal memperbarui data kamar."); // Pesan jika terjadi
error
    e.printStackTrace(); // Menampilkan detail error untuk debugging
}
}

```

```

try (PreparedStatement stmt = koneksi.prepareStatement(sql)) {
    stmt.setInt(1, nomorHapus);
}

```

```

        stmt.executeUpdate();
        System.out.println("Data kamar dengan nomor " + nomorHapus + "
berhasil dihapus.");
    } catch (SQLException e) { // Menangkap exception jika ada kesalahan
dalam eksekusi SQL
        System.out.println("Gagal menghapus data kamar.");
        e.printStackTrace();//Menampilkan rincian kesalahan (stack trace) untuk
membantu debugging jika terjadi masalah saat eksekusi query
    }
}

```

```

public static void tanggal(Scanner scanner) {
    try {
        System.out.print("Masukkan nomor kamar: ");
        int nomorKamar = scanner.nextInt();
        scanner.nextLine(); // Bersihkan buffer input

        if (!cekNomorKamar(nomorKamar)) { //Percabangan (if statement)
            System.out.println("Nomor kamar tidak ditemukan. Silakan masukkan
nomor yang valid.");
            return;
        }

        System.out.print("Masukkan tanggal check-in (dd-MM-yyyy): ");
        String tanggalCheckIn = scanner.nextLine(); // Gunakan nextLine untuk
membaca input
        System.out.print("Masukkan jumlah malam menginap: ");
        int jumlahMalam = scanner.nextInt();
        scanner.nextLine(); // Bersihkan buffer input setelah membaca angka

        SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
        Date date = format.parse(tanggalCheckIn); // Mengubah string tanggal
menjadi objek Date

        Calendar calendar = Calendar.getInstance(); // Membuat objek Calendar
yang dapat digunakan untuk manipulasi tanggal
        calendar.setTime(date); // Menetapkan tanggal check-in ke objek calendar
        calendar.add(Calendar.DAY_OF_MONTH, jumlahMalam); // perhitungan
matematis terkait menambahkan jumlah malam yang diinputkan oleh pengguna
ke tanggal check-in

        Date tanggalCheckOut = calendar.getTime(); // Mendapatkan tanggal
check-out sebagai objek Date
        System.out.println("Nomor kamar: " + nomorKamar);
        System.out.println("Tanggal check-in: " + format.format(date));
    }
}

```

```

        System.out.println("Tanggal check-out: " +
format.format(tanggalCheckOut));

    } catch (ParseException e) {
        // Menangani kesalahan jika format tanggal tidak valid
        System.out.println("Format tanggal tidak valid. Gunakan format dd-MM-
yyyy.");
    } catch (InputMismatchException e) {
        // Menangani kesalahan jika input bukan angka yang diharapkan
        System.out.println("Input tidak valid.");
        scanner.nextLine(); // Bersihkan buffer input jika terjadi kesalahan
    }
}

```

➤ Collection Framework

```
ArrayList<Kamar> daftarKamar; // Daftar kamar menggunakan ArrayList
```

ArrayList diinisialisasi dalam konstruktor kelas Hotel untuk memastikan siap digunakan.

```
this.daftarKamar = new ArrayList<>(); // Inisialisasi ArrayList
```

ArrayList diinisialisasi dalam konstruktor kelas Hotel untuk memastikan siap digunakan.

```
daftarKamar.add(kamar);
```

Menambahkan objek Kamar ke dalam ArrayList. Ini memanfaatkan metode bawaan dari Collection Framework.

➤ Menggunakan JDBC dan Fungsi Create, Read, Update, Delete (CRUD)

• JDBC

```
private static Connection koneksi; // Variabel statik untuk menyimpan koneksi
database
```

```
private static final String PASSWORD = "admin123"; // Password yang harus
dimasukkan
```

```

// Method untuk membuat koneksi ke database
@SuppressWarnings("CallToPrintStackTrace")
public static void koneksiDatabase() {
    try { //Exception Handling
        // Memuat driver JDBC untuk MySQL
        Class.forName("com.mysql.cj.jdbc.Driver"); // Constructor: Memuat driver
untuk koneksi
    }
}

```

```

// String: URL koneksi ke database
String url =
"jdbc:mysql://localhost:3306/hotel_db?useSSL=false&serverTimezone=UTC";
String user = "root"; // String: Username database
String password = ""; // String: Password database

// Membuka koneksi ke database
koneksi = DriverManager.getConnection(url, user, password); // Constructor:
Membuat koneksi menggunakan DriverManager
System.out.println("Koneksi ke database berhasil."); // String: Pesan sukses
} catch (ClassNotFoundException | SQLException e) { // Exception Handling:
Menangkap kesalahan
System.out.println("Koneksi ke database gagal."); // String: Pesan error
e.printStackTrace(); // Menampilkan error stack trace untuk debugging
System.exit(0); // Percabangan: Keluar dari program jika koneksi gagal
}
}

```

- **Create**

```

// Fungsi Create: Menambah data kamar ke database
@SuppressWarnings("CallToPrintStackTrace")
public static void tambahKamar(int nomorKamar, String tipeKamar, double
hargaPerMalam) {
// Query SQL untuk memeriksa apakah nomor kamar sudah ada
String checkSql = "SELECT 1 FROM kamar WHERE nomor_kamar = ?";

try (PreparedStatement checkStmt = koneksi.prepareStatement(checkSql)) {
// Exception handling: Membuka statement SQL untuk pengecekan
checkStmt.setInt(1, nomorKamar); // Mengatur parameter query untuk
nomor kamar
ResultSet rs = checkStmt.executeQuery(); // Menjalankan query dan
menyimpan hasilnya

// Percabangan: Memeriksa apakah nomor kamar sudah ada di database
if (rs.next()) {
System.out.println("Nomor kamar sudah ada, tidak ditambahkan."); //
Pesan jika nomor kamar sudah ada
return; // Keluar dari metode tanpa menambahkan data baru
}

// Query SQL untuk menambahkan data kamar baru ke tabel
String sql = "INSERT INTO kamar (nomor_kamar, tipe_kamar,
harga_per_malam, tersedia) VALUES (?, ?, ?, ?)";
try (PreparedStatement stmt = koneksi.prepareStatement(sql)) {
// Mengatur parameter untuk query INSERT

```



```

        stmt.setInt(1, nomorKamar); // Parameter nomor kamar
        stmt.setString(2, tipeKamar); // Parameter tipe kamar
        stmt.setDouble(3, hargaPerMalam); // Parameter harga per malam
        stmt.setBoolean(4, true); // Parameter tersedia (default: true)

        stmt.executeUpdate(); // Menjalankan query untuk menambahkan data
        System.out.println("Data kamar berhasil ditambahkan."); // Pesan
sukses
    } catch (SQLException e) { // Exception handling untuk query INSERT
        System.out.println("Gagal menambahkan data kamar."); // Pesan error
        e.printStackTrace(); // Menampilkan rincian error untuk debugging
    }
    } catch (SQLException e) { // Exception handling untuk pengecekan data
        System.out.println("Gagal memeriksa data kamar."); // Pesan error
        e.printStackTrace(); // Menampilkan rincian error untuk debugging
    }
}
}

```

- **Read**

```

// Fungsi Read: Membaca data kamar dari database
@SuppressWarnings("CallToPrintStackTrace")
public static void bacaKamar() {
    // Query SQL untuk membaca semua data dari tabel 'kamar'
    String sql = "SELECT nomor_kamar, tipe_kamar, harga_per_malam, tersedia
FROM kamar";

    try (Statement stmt = koneksi.createStatement(); ResultSet rs =
stmt.executeQuery(sql)) {
        // Exception handling: Membuka statement dan eksekusi query
        // Perulangan: Mengiterasi hasil query menggunakan while loop
        while (rs.next()) {
            // Menampilkan data kamar satu per satu
            System.out.println("Nomor Kamar: " + rs.getInt("nomor_kamar")); //
Menampilkan nomor kamar
            System.out.println("Tipe Kamar: " + rs.getString("tipe_kamar")); //
Menampilkan tipe kamar
            System.out.println("Harga per Malam: Rp " +
rs.getDouble("harga_per_malam")); // Menampilkan harga per malam

            // Percabangan: Menentukan apakah kamar tersedia atau tidak
            System.out.println("Tersedia: " + (rs.getBoolean("tersedia") ? "Ya" :
"Tidak"));
            System.out.println("-----");
        }
    }
}

```

```

    } catch (SQLException e) { // Exception handling: Menangkap kesalahan
    saat eksekusi query SQL
        System.out.println("Gagal membaca data kamar."); // Menampilkan pesan
    error
        e.printStackTrace(); // Menampilkan rincian error untuk debugging
    }
}

```

- **Update**

```

// Fungsi Update: Memperbarui data kamar di database
@SuppressWarnings("CallToPrintStackTrace")
public static void updateKamar(Scanner scanner) {
    // Menampilkan daftar kamar yang tersedia
    bacaKamar();

    // Meminta pengguna memasukkan nomor kamar yang akan diperbarui
    System.out.print("Masukkan nomor kamar yang ingin diperbarui: ");
    int nomorKamar = scanner.nextInt();
    scanner.nextLine(); // Membersihkan buffer input

    // Percabangan: Memeriksa apakah nomor kamar valid
    if (!cekNomorKamar(nomorKamar)) {
        System.out.println("Nomor kamar tidak ditemukan."); // Pesan jika nomor
    kamar tidak ditemukan
        return; // Keluar dari metode jika nomor kamar tidak valid
    }

    // Meminta data baru untuk memperbarui kamar
    System.out.print("Masukkan tipe kamar baru: ");
    String tipeBaru = scanner.nextLine(); // Membaca tipe kamar baru
    System.out.print("Masukkan harga baru per malam: ");
    double hargaBaru = scanner.nextDouble(); // Membaca harga kamar baru
    scanner.nextLine(); // Membersihkan buffer input
}

```

```

// Query SQL untuk memperbarui data kamar

String sql = "UPDATE kamar SET tipe_kamar = ?, harga_per_malam = ?
WHERE nomor_kamar = ?";

try (PreparedStatement stmt = koneksi.prepareStatement(sql)) { // Exception
handling: Membuka koneksi SQL

    // Mengatur parameter query SQL

    stmt.setString(1, tipeBaru); // Parameter tipe kamar

    stmt.setDouble(2, hargaBaru); // Parameter harga kamar

    stmt.setInt(3, nomorKamar); // Parameter nomor kamar


    // Menjalankan query update

    stmt.executeUpdate();

    System.out.println("Data kamar berhasil diperbarui."); // Pesan sukses
} catch (SQLException e) { // Exception handling: Menangkap kesalahan
SQL

    System.out.println("Gagal memperbarui data kamar."); // Pesan jika terjadi
error

    e.printStackTrace(); // Menampilkan detail error untuk debugging

}
}

```

- **Delete**

```

// Fungsi Delete: Menghapus data kamar dari database
@SuppressWarnings("CallToPrintStackTrace")
public static void hapusKamar(int nomorHapus) {
    String sql = "DELETE FROM kamar WHERE nomor_kamar = ?";
    //Menggunakan try-with-resources untuk memastikan statement ditutup
secara otomatis setelah selesai.
    try (PreparedStatement stmt = koneksi.prepareStatement(sql)) {
        stmt.setInt(1, nomorHapus);
        stmt.executeUpdate();
        System.out.println("Data kamar dengan nomor " + nomorHapus + "
berhasil dihapus.");
    } catch (SQLException e) { // Menangkap exception jika ada kesalahan
dalam eksekusi SQL

```

```
        System.out.println("Gagal menghapus data kamar.");
        e.printStackTrace();//Menampilkan rincian kesalahan (stack trace) untuk
        membantu debugging jika terjadi masalah saat eksekusi query
    }
}
```

C. CONSOLE

```
C:\TBBARU> c: && cd c:\TBBARU && cmd /C ""C:\Program Files\Eclipse
Adoptium\jdk-17.0.12.7-hotspot\bin\java.exe"
@C:\Users\HP\AppData\Local\Temp\cp_54y0i3p0mkow1cd8522xnpjyh.argfile Main "
```

Koneksi ke database berhasil.

Masukkan password untuk mengakses sistem: admin123

Password benar

Selamat datang di sistem Manajemen hotel

=== Verifikasi CAPTCHA ===

Berapa hasil dari: 29 + 39?

Jawaban Anda: 68

CAPTCHA benar

Selamat datang!

=== Manajemen Kamar Hotel ===

1. Tambah Kamar
2. Lihat Data Kamar
3. Perbarui Data Kamar
4. Hapus Data Kamar
5. Tanggal Check-in dan Check-out
6. Keluar

Pilih menu: 1

Masukkan nomor kamar: 28

Masukkan tipe kamar: double

Masukkan harga per malam: 400000

Data kamar berhasil ditambahkan.

=== Manajemen Kamar Hotel ===

1. Tambah Kamar
2. Lihat Data Kamar
3. Perbarui Data Kamar
4. Hapus Data Kamar
5. Tanggal Check-in dan Check-out
6. Keluar

Pilih menu: 2

Nomor Kamar: 2

Tipe Kamar: double

Harga per Malam: Rp 300000.0

Tersedia: Ya

Nomor Kamar: 3

Tipe Kamar: single

Harga per Malam: Rp 300000.0

Tersedia: Ya

Nomor Kamar: 8

Tipe Kamar: Single

Harga per Malam: Rp 250000.0

Tersedia: Ya

Nomor Kamar: 29

Tipe Kamar: single

Harga per Malam: Rp 500000.0

Tersedia: Ya

Nomor Kamar: 101

Tipe Kamar: Single

Harga per Malam: Rp 250000.0

Tersedia: Ya

Nomor Kamar: 123

Tipe Kamar: suite

Harga per Malam: Rp 400000.0

Tersedia: Ya

Nomor Kamar: 234

Tipe Kamar: double

Harga per Malam: Rp 300000.0

Tersedia: Ya

Nomor Kamar: 345

Tipe Kamar: DOUBLE

Harga per Malam: Rp 400000.0

Tersedia: Ya

=== Manajemen Kamar Hotel ===

1. Tambah Kamar
 2. Lihat Data Kamar
 3. Perbarui Data Kamar
 4. Hapus Data Kamar
 5. Tanggal Check-in dan Check-out
 6. Keluar
- Pilih menu: 3

Nomor Kamar: 2
Tipe Kamar: double
Harga per Malam: Rp 300000.0
Tersedia: Ya

Nomor Kamar: 3
Tipe Kamar: single
Harga per Malam: Rp 300000.0
Tersedia: Ya

Nomor Kamar: 8
Tipe Kamar: Single
Harga per Malam: Rp 250000.0
Tersedia: Ya

Nomor Kamar: 29
Tipe Kamar: single
Harga per Malam: Rp 500000.0
Tersedia: Ya

Nomor Kamar: 101
Tipe Kamar: Single
Harga per Malam: Rp 250000.0
Tersedia: Ya

Nomor Kamar: 123
Tipe Kamar: suite
Harga per Malam: Rp 400000.0
Tersedia: Ya

Nomor Kamar: 234
Tipe Kamar: double
Harga per Malam: Rp 300000.0
Tersedia: Ya

Nomor Kamar: 345
Tipe Kamar: DOUBLE
Harga per Malam: Rp 400000.0
Tersedia: Ya

Masukkan nomor kamar yang ingin diperbarui: 345
Masukkan tipe kamar baru: suite
Masukkan harga baru per malam: 500000
Data kamar berhasil diperbarui.

=== Manajemen Kamar Hotel ===

1. Tambah Kamar
2. Lihat Data Kamar
3. Perbarui Data Kamar
4. Hapus Data Kamar
5. Tanggal Check-in dan Check-out
6. Keluar

Pilih menu: 4

Masukkan nomor kamar yang ingin dihapus: 234

Data kamar dengan nomor 234 berhasil dihapus.

=== Manajemen Kamar Hotel ===

1. Tambah Kamar
2. Lihat Data Kamar
3. Perbarui Data Kamar
4. Hapus Data Kamar
5. Tanggal Check-in dan Check-out
6. Keluar

Pilih menu: 5

Masukkan nomor kamar: 29

Masukkan tanggal check-in (dd-MM-yyyy): 28-12-2004

Masukkan jumlah malam menginap: 2

Nomor kamar: 29

Tanggal check-in: 28-12-2004

Tanggal check-out: 30-12-2004

=== Manajemen Kamar Hotel ===

1. Tambah Kamar
2. Lihat Data Kamar
3. Perbarui Data Kamar
4. Hapus Data Kamar
5. Tanggal Check-in dan Check-out
6. Keluar

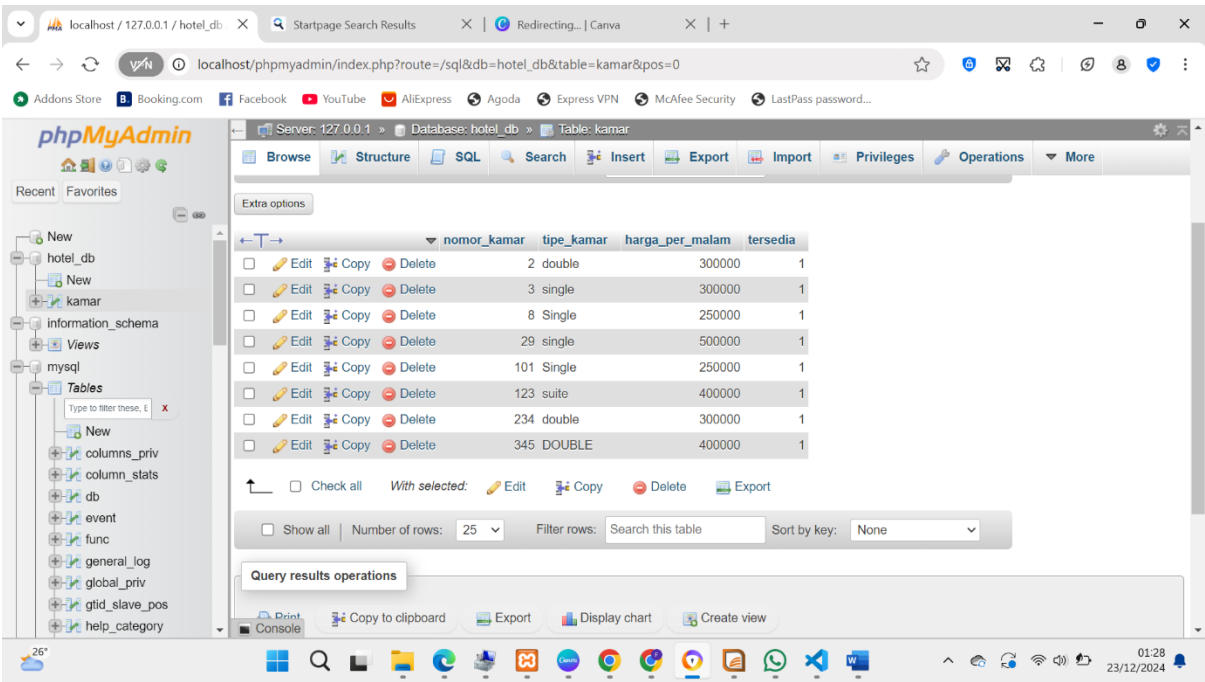
Pilih menu: 6

Terima kasih telah menggunakan sistem management hotel.

Koneksi database ditutup.

c:\TBBARU>

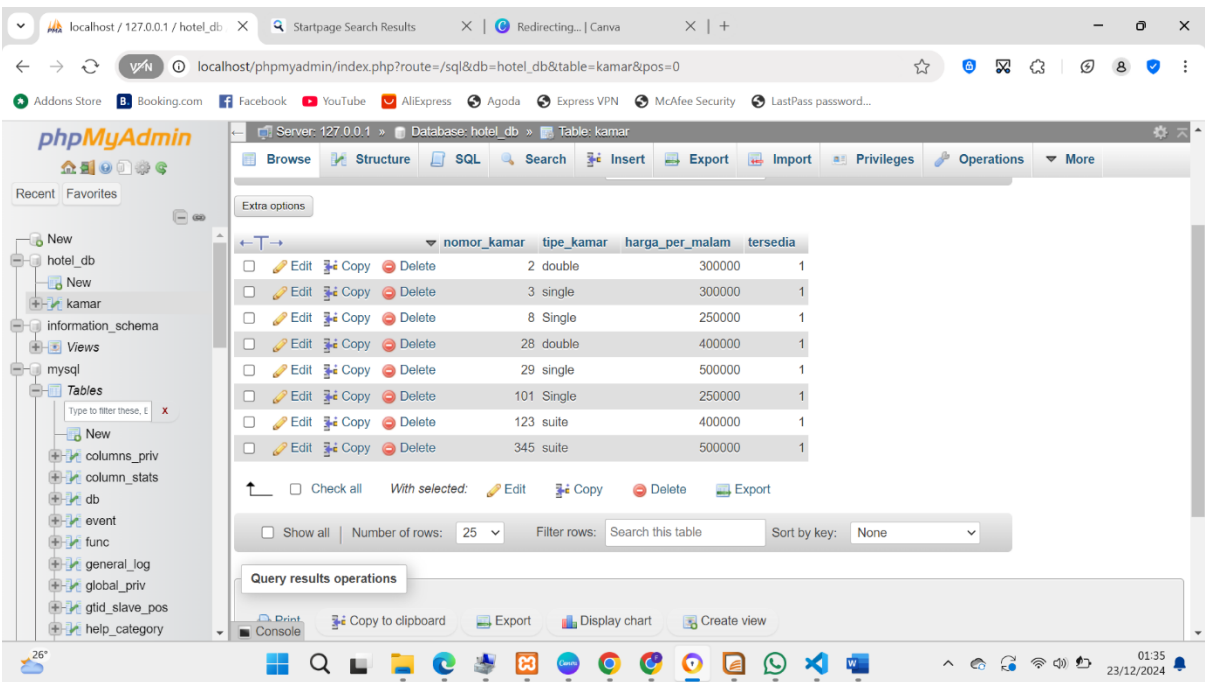
Data sebelum diperbarui



The screenshot shows the phpMyAdmin interface for the 'hotel_db' database. The 'kamar' table is selected, and the 'Browse' tab is active. The table contains 8 rows of data. The columns are 'nomor_kamar', 'tipe_kamar', 'harga_per_malam', and 'tersedia'.

nomor_kamar	tipe_kamar	harga_per_malam	tersedia
2	double	300000	1
3	single	300000	1
8	Single	250000	1
29	single	500000	1
101	Single	250000	1
123	suite	400000	1
234	double	300000	1
345	DOUBLE	400000	1

Data setelah diperbarui



The screenshot shows the phpMyAdmin interface for the 'hotel_db' database. The 'kamar' table is selected, and the 'Browse' tab is active. The table contains 8 rows of data. The columns are 'nomor_kamar', 'tipe_kamar', 'harga_per_malam', and 'tersedia'.

nomor_kamar	tipe_kamar	harga_per_malam	tersedia
2	double	300000	1
3	single	300000	1
8	Single	250000	1
28	double	400000	1
29	single	500000	1
101	Single	250000	1
123	suite	400000	1
345	suite	500000	1