# PENDAHULUAN

## Deskripsi Umum

Studi kasus ini fokus pada pengembangan sistem manajemen lahan parkir yang bertujuan untuk meningkatkan efisiensi dan pengalaman pengguna. Sistem tersebut mencakup teknologi sensor pintar untuk mendeteksi ketersediaan tempat parkir, aplikasi seluler untuk pengguna agar dapat melihat dan memesan tempat parkir, serta integrasi dengan sistem pembayaran elektronik. Dengan implementasi ini, tujuan utama adalah mengurangi waktu mencari tempat parkir, meningkatkan penggunaan lahan parkir, dan memperkenalkan metode pembayaran yang lebih praktis. Studi kasus ini dapat menjadi acuan bagi kota-kota yang ingin meningkatkan manajemen lahan parkir mereka.

## Konsep Object Oriented Programming yang Digunakan

Pemrograman Berorientasi Objek (OOP) adalah paradigma pemrograman yang memandang program sebagai kumpulan objek yang saling berinteraksi. Objek merupakan instansi dari kelas, yang memiliki atribut (data) dan metode (fungsi) yang dapat memanipulasi data tersebut. OOP membantu mengorganisasi dan memahami kompleksitas kode dengan memfasilitasi konsep pewarisan, enkapsulasi, dan polimorfisme, memungkinkan pembangunan perangkat lunak yang lebih modular, mudah dimengerti, dan dapat diubah.

# ANALISIS PERSIAPAN

## Topik Bahasan

1. Modul 1:Pengenalan Java OOP
2. Modul 2:Class Dan Bagianya
3. Modul 3:Constructor Dan Object

## Tujuan Pertemuan

Pada pertemuan 1 ini saya mempelajari :

• Menyajikan konsep dasar Pemrograman Berorientasi Objek (OOP) dalam bahasa pemrograman Java.

• Memperkenalkan prinsip-prinsip utama seperti class, object, inheritance, polymorphism, encapsulation, dan abstraction.

• Mendetailkan struktur dan sintaksis pembuatan class dalam Java.

• Membahas komponen-komponen kelas seperti atribut

• Menjelaskan konsep konstruktor untuk inisialisasi objek.

• Membahas pembuatan dan penggunaan objek dalam konteks pemrograman berorientasi objek dengan menggunakan Java.

## Isi Pertemuan

|  |
| --- |
| private String username;  private String password;  public Admin(String username, String password) {  this.username = username;  this.password = password;  } |

**Penjelasan**

Program ini merupakan bagian dari implementasi sebuah kelas "Admin" dalam pemrograman berorientasi objek (OOP). Kelas ini memiliki dua atribut private yaitu "username" dan "password" yang tidak dapat diakses langsung dari luar kelas. Constructor "Admin" digunakan untuk menginisialisasi objek admin dengan memberikan nilai awal pada atribut "username" dan "password" berdasarkan parameter yang diterima.

|  |
| --- |
| class Kendaraan {  private String platNomor;  private Date jamMasuk;  public Kendaraan(String platNomor) {  this.platNomor = platNomor;  this.jamMasuk = new Date();  } |

**Penjelasan**

Kode tersebut adalah awal dari implementasi kelas "Kendaraan" dalam pemrograman berorientasi objek (OOP) menggunakan bahasa Java. Kelas ini memiliki dua atribut private, yaitu "platNomor" dan "jamMasuk", yang menyimpan informasi plat nomor kendaraan dan waktu masuknya.

Dalam konstruktor "Kendaraan", parameter "platNomor" yang diterima saat pembuatan objek diassign ke atribut "platNomor", sementara atribut "jamMasuk" diinisialisasi dengan nilai waktu saat itu menggunakan objek dari kelas "Date". Dengan cara ini, setiap kali objek "Kendaraan" dibuat, waktu masuk kendaraan otomatis dicatat.

# IMPLEMENTASI

## Topik Bahasan

1. Modul 4:Input Process Output
2. Modul 5:Data Collection

## Tujuan Pertemuan

Disini saya mendapatkan konsep dasar dalam pemrosesan informasi, mengenai bagaimana data dimasukkan ke dalam sistem, diolah, dan dihasilkan sebagai output. Modul tentang pengumpulan data mungkin mencakup berbagai metode dan teknik untuk mengumpulkan informasi secara efektif. Pastikan untuk merinci materi atau catatan yang diberikan selama pertemuan untuk mendapatkan gambaran yang lebih jelas tentang pembelajaran Anda.

## Hasil Implementasi

|  |
| --- |
| public boolean login() {  Scanner scanner = new Scanner(System.in);  System.out.println("Login Admin");  System.out.print("Username: ");  String enteredUsername = scanner.next();  System.out.print("Password: ");  String enteredPassword = scanner.next();  if (admin.authenticate(enteredUsername, enteredPassword)) {  System.out.println("Login berhasil!");  return true;  } else {  System.out.println("Login gagal. Coba lagi.");  return false;  }  } |

**Penjelasan**

Program ini adalah bagian dari suatu sistem login yang meminta pengguna untuk memasukkan username dan password. Berikut adalah penjelasan singkat dari setiap bagian program tersebut:

1. Scanner dan Input:

Menggunakan Scanner untuk mendapatkan input dari pengguna.

Menampilkan pesan "Login Admin" untuk memberikan petunjuk penggunaan.

1. Meminta Input Username dan Password:

Menampilkan pesan "Username: " dan "Password: " untuk meminta pengguna memasukkan informasi login.

Menggunakan scanner.next() untuk membaca input dari pengguna dan menyimpannya di variabel enteredUsername dan enteredPassword.

1. Authentication:

Menggunakan objek admin (yang harus dideklarasikan sebelumnya dalam program) untuk melakukan autentikasi.

Memanggil metode authenticate pada objek admin dengan meneruskan input username dan password yang dimasukkan pengguna.

1. Output Login:

Jika autentikasi berhasil, menampilkan pesan "Login berhasil!" dan mengembalikan nilai true.

Jika autentikasi gagal, menampilkan pesan "Login gagal. Coba lagi." dan mengembalikan nilai false.

|  |
| --- |
| class ParkirManager {  private ArrayList<Kendaraan> dataKendaraan;  private ArrayList<Kendaraan> historisKendaraan;  public ParkirManager() {  this.dataKendaraan = new ArrayList<>();  this.historisKendaraan = new ArrayList<>();  loadDaftarKendaraanHistoris();  } |

**Penjelasan**

Kode tersebut adalah awal dari implementasi kelas ParkirManager dalam pemrograman berorientasi objek (OOP) menggunakan bahasa Java. Berikut adalah penjelasan singkatnya:

Variabel Kelas:

dataKendaraan: ArrayList yang menyimpan daftar kendaraan yang saat ini berada di area parkir. Ini digunakan untuk melacak kendaraan yang masih berada di tempat parkir.

historisKendaraan: ArrayList yang menyimpan daftar kendaraan yang telah meninggalkan area parkir. Ini digunakan untuk menyimpan riwayat kendaraan yang sudah parkir sebelumnya.

Konstruktor:

Konstruktor kelas ParkirManager inisialisasi dua ArrayList, yaitu dataKendaraan dan historisKendaraan.

Memanggil metode loadDaftarKendaraanHistoris(), yang kemungkinan akan digunakan untuk mengambil data historis kendaraan dari database atau sumber data lainnya dan memuatnya ke dalam ArrayList historisKendaraan.

## Topik Bahasan

1. Modul 6: Encapsulation
2. Modul 7:Inheritance

## Tujuan Pertemuan

Anda mungkin telah mempelajari dua konsep kunci dalam pemrograman berorientasi objek (OOP): inheritance (warisan) dan encapsulation (enkapsulasi).

1. Inheritance:

Mewarisi sifat-sifat (atribut, metode) dari kelas yang sudah ada.

Membantu mengorganisir dan mengelompokkan kode, mengurangi duplikasi, memungkinkan perluasan fungsionalitas.

1. Encapsulation:

Pembungkusan atribut dan metode dalam satu unit (kelas).

Mengatur akses ke bagian-bagian tertentu dari kelas, meningkatkan keamanan data, dan menyembunyikan implementasi internal dari pengguna luar.

Kedua konsep ini bersama-sama membentuk dasar untuk membangun struktur kode yang efisien dan mudah dimengerti dalam paradigma pemrograman berorientasi objek.

## Hasil Implementasi

|  |
| --- |
| class ParkirManager {  private ArrayList<Kendaraan> dataKendaraan;  private ArrayList<Kendaraan> historisKendaraan;  public ParkirManager() {  this.dataKendaraan = new ArrayList<>();  this.historisKendaraan = new ArrayList<>();  loadDaftarKendaraanHistoris(); // Load historis dari database  }  // 3. Polymorphism  public Kendaraan checkIn(String platNomor) {  Kendaraan kendaraan = new Mobil(platNomor);  dataKendaraan.add(kendaraan);  saveDaftarKendaraanHistoris(); // Simpan historis ke database  return kendaraan;  }  public void checkOut(Kendaraan kendaraan) {  // Menghitung biaya parkir  long durasi = new Date().getTime() - kendaraan.getJamMasuk().getTime();  long biaya = hitungBiayaParkir(durasi);  // Mencetak detail struk  cetakStruk(kendaraan, biaya);  // Memindahkan kendaraan dari dataKendaraan ke historisKendaraan  historisKendaraan.add(kendaraan);  dataKendaraan.remove(kendaraan);  // Simpan perubahan ke database  saveDaftarKendaraanHistoris();  } |

**Penjelasan**

Program ini adalah implementasi kelas ParkirManager dalam pemrograman berorientasi objek (OOP) menggunakan Java. Kelas ini memiliki variabel untuk menyimpan daftar kendaraan yang sedang parkir dan historis kendaraan yang telah meninggalkan area parkir. Metode checkIn menambahkan kendaraan baru ke dalam daftar parkir, sedangkan checkOut menghitung biaya parkir, mencetak struk, dan memindahkan kendaraan ke historis. Program ini juga mencatat perubahan ke database untuk memastikan data tetap konsisten.

|  |
| --- |
| class Kendaraan {  private String platNomor;  private Date jamMasuk;  public Kendaraan(String platNomor) {  this.platNomor = platNomor;  this.jamMasuk = new Date();  }  public String getPlatNomor() {  return platNomor;  }  public Date getJamMasuk() {  return jamMasuk;  }  }  class Mobil extends Kendaraan {  public Mobil(String platNomor) {  super(platNomor);  }  } |

**Penjelasan**

Program ini mengimplementasikan dua kelas dalam pemrograman berorientasi objek (OOP) menggunakan bahasa Java: Kendaraan dan Mobil. Berikut adalah penjelasan singkatnya:

1. Kelas Kendaraan:

Variabel Anggota:

platNomor: String, menyimpan nomor plat kendaraan.

jamMasuk: Objek Date, menyimpan waktu masuk kendaraan.

1. Konstruktor:

Menerima parameter platNomor dan menginisialisasi platNomor dengan nilai tersebut.

Menginisialisasi jamMasuk dengan waktu saat objek dibuat.

1. Metode Get:

getPlatNomor(): Mengembalikan nilai platNomor.

getJamMasuk(): Mengembalikan nilai jamMasuk.

Kelas Mobil (Turunan dari Kendaraan):

1. Konstruktor:

Menerima parameter platNomor dan memanggil konstruktor kelas induk (Kendaraan) menggunakan super(platNomor).

Menginisialisasi objek Mobil dengan nomor plat yang diberikan.

Kelas Mobil mewarisi sifat-sifat dari kelas Kendaraan. Dengan demikian, setiap objek Mobil adalah objek Kendaraan yang juga memiliki nomor plat dan waktu masuk. Konsep ini memanfaatkan hubungan warisan (inheritance) dalam OOP, memungkinkan kelas turunan (Mobil) memanfaatkan atribut dan metode dari kelas induk (Kendaraan).

## Topik Bahasan

1. Modul 8: …
2. Modul 9: …

## Tujuan Pertemuan

Tulis apa yang Anda dapat / pelajari pada pertemuan ini…

## Hasil Implementasi

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class / Fungsi

**Penjelasan**

Tulis Penjelasan disini …

## Topik Bahasan

1. Modul 10: …
2. Modul 11: …

## Tujuan Pertemuan

Tulis apa yang Anda dapat / pelajari pada pertemuan ini…

## Hasil Implementasi

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class / Fungsi

**Penjelasan**

Tulis Penjelasan disini …

## Topik Bahasan

1. Modul 12: …
2. Modul 13: …

## Tujuan Pertemuan

Tulis apa yang Anda dapat / pelajari pada pertemuan ini…

## Hasil Implementasi

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class / Fungsi

**Penjelasan**

Tulis Penjelasan disini …

## Topik Bahasan

1. Modul 14: …

## Tujuan Pertemuan

Tulis apa yang Anda dapat / pelajari pada pertemuan ini…

## Hasil Implementasi

Tulis kode program dikotak ini...

1 kotak dan 1 Penjelasan untuk 1 Class / Fungsi

**Penjelasan**

Tulis Penjelasan disini …

# HASIL DAN DOKUMENTASI

Berisi tampilan Output GUI lengkap dengan penjelasan alur pemakaian aplikasinya.