LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
PERTEMUAN 6
STUDI KASUS

Disusun oleh:

Nama	: Aditya Lucky Zulkarnaen
NIM	: 24/537764/SV/24449
Kelas	: PLB1
Dosen Pengampu	: Margareta Hardiyanti, S.Kom., M.IM., M.Cs

PROGRAM STUDI D-IV TEKNOLOGI REKAYASA PERANGKAT LUNAK
DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA

2025

PERTEMUAN 6

STUDI KASUS

A. Sistem Management Parkir

1. Copy code

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;
import java.time.format.DateTimeFormatter;

class Kendaraan {
    private String nomorPlat;
    private int jumlahRoda;
    private LocalDateTime waktuMasuk;

    public Kendaraan(String nomorPlat, int jumlahRoda) {
        this.nomorPlat = nomorPlat;
        this.jumlahRoda = jumlahRoda;
        this.waktuMasuk = LocalDateTime.now();
    }

    public String getNomorPlat() {
        return nomorPlat;
    }

    public int getJumlahRoda() {
        return jumlahRoda;
    }
```

```
    public LocalDateTime getWaktuMasuk() {
        return waktuMasuk;
    }

    public String getJenisKendaraan() {
        return (jumlahRoda == 2) ? "Motor" : "Mobil";
    }

    @Override
    public String toString() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy
HH:mm:ss");
        return getJenisKendaraan() + " - Plat: " + nomorPlat +
                " | Roda: " + jumlahRoda +
                " | Masuk: " + waktuMasuk.format(formatter);
    }
}

class Lantai {
    private int nomorLantai;
    private List<Kendaraan> daftarKendaraan;
    private final int KAPASITAS_RODA = 12;

    public Lantai(int nomorLantai) {
        this.nomorLantai = nomorLantai;
        this.daftarKendaraan = new ArrayList<>();
    }

    public int getNomorLantai() {
        return nomorLantai;
    }
```

```
public List<Kendaraan> getDaftarKendaraan() {
    return daftarKendaraan;
}

public int getJumlahRodaSaatIni() {
    int totalRoda = 0;
    for (Kendaraan kendaraan : daftarKendaraan) {
        totalRoda += kendaraan.getJumlahRoda();
    }
    return totalRoda;
}

public int getSisaKapasitasRoda() {
    return KAPASITAS_RODA - getJumlahRodaSaatIni();
}

public boolean tambahKendaraan(Kendaraan kendaraan) {
    if (kendaraan.getJumlahRoda() <= getSisaKapasitasRoda()) {
        daftarKendaraan.add(kendaraan);
        return true;
    }
    return false;
}

public Kendaraan keluarkanKendaraan(String nomorPlat) {
    for (int i = 0; i < daftarKendaraan.size(); i++) {
        Kendaraan kendaraan = daftarKendaraan.get(i);
        if (kendaraan.getNomorPlat().equalsIgnoreCase(nomorPlat)) {
            daftarKendaraan.remove(i);
            return kendaraan;
```

```
            }
        }
        return null;
    }

    public Kendaraan cariKendaraan(String nomorPlat) {
        for (Kendaraan kendaraan : daftarKendaraan) {
            if (kendaraan.getNomorPlat().equalsIgnoreCase(nomorPlat)) {
                return kendaraan;
            }
        }
        return null;
    }
}

class SistemParkir {
    private List<Lantai> daftarLantai;
    private final int BIAYA_PARKIR_PER_JAM_MOTOR = 2000;
    private final int BIAYA_PARKIR_PER_JAM_MOBIL = 5000;

    public SistemParkir(int jumlahLantai) {
        this.daftarLantai = new ArrayList<>();
        for (int i = 1; i <= jumlahLantai; i++) {
            daftarLantai.add(new Lantai(i));
        }
    }

    public void tampilkanDaftarKendaraan() {
        System.out.println("\n===== DAFTAR KENDARAAN PARKIR =====");
        for (Lantai lantai : daftarLantai) {
            System.out.println("Lantai " + lantai.getNomorLantai() + ":");
```

```
        System.out.println("Jumlah roda saat ini: " + lantai.getJumlahRodaSaatIni() +
                           " | Sisa kapasitas: " + lantai.getSisaKapasitasRoda() + " roda");

        List<Kendaraan> kendaraanList = lantai.getDaftarKendaraan();
        if (kendaraanList.isEmpty()) {
            System.out.println("- Tidak ada kendaraan");
        } else {
            for (int i = 0; i < kendaraanList.size(); i++) {
                System.out.println((i+1) + ". " + kendaraanList.get(i));
            }
        }
        System.out.println();
    }
}

public boolean tambahKendaraan(Scanner scanner) {
    System.out.println("\n===== TAMBAH KENDARAAN =====");
    System.out.print("Masukkan nomor plat: ");
    String nomorPlat = scanner.nextLine();

    System.out.println("Jenis kendaraan:");
    System.out.println("1. Motor (2 roda)");
    System.out.println("2. Mobil (4 roda)");
    System.out.print("Pilih jenis kendaraan (1/2): ");

    int jenisKendaraan;
    try {
        jenisKendaraan = Integer.parseInt(scanner.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Input tidak valid!");
        return false;
```

```
}

int jumlahRoda;
if (jenisKendaraan == 1) {
   jumlahRoda = 2;
} else if (jenisKendaraan == 2) {
   jumlahRoda = 4;
} else {
   System.out.println("Pilihan tidak valid!");
   return false;
}

System.out.print("Masukkan nomor lantai (1-" + daftarLantai.size() + "): ");
int nomorLantai;
try {
   nomorLantai = Integer.parseInt(scanner.nextLine());
} catch (NumberFormatException e) {
   System.out.println("Input tidak valid!");
   return false;
}

if (nomorLantai < 1 || nomorLantai > daftarLantai.size()) {
   System.out.println("Nomor lantai tidak valid!");
   return false;
}

Lantai lantai = daftarLantai.get(nomorLantai - 1);

for (Lantai l : daftarLantai) {
   if (l.cariKendaraan(nomorPlat) != null) {
```

```
            System.out.println("Kendaraan dengan nomor plat " + nomorPlat + " sudah terparkir!");
            return false;
        }
    }

    Kendaraan kendaraan = new Kendaraan(nomorPlat, jumlahRoda);

    if (lantai.tambahKendaraan(kendaraan)) {
        System.out.println(kendaraan.getJenisKendaraan() + " dengan nomor plat " + nomorPlat +
                           " berhasil diparkir di lantai " + nomorLantai);
        return true;
    } else {
        System.out.println("Lantai " + nomorLantai + " tidak cukup ruang untuk kendaraan ini!");
        return false;
    }
  }

  public boolean pindahkanKendaraan(Scanner scanner) {
    System.out.println("\n===== PINDAHKAN KENDARAAN =====");
    System.out.print("Masukkan nomor plat kendaraan yang akan dipindahkan: ");
    String nomorPlat = scanner.nextLine();

    Lantai lantaiAsal = null;
    Kendaraan kendaraan = null;

    for (Lantai lantai : daftarLantai) {
        Kendaraan k = lantai.cariKendaraan(nomorPlat);
        if (k != null) {
```

```
            lantaiAsal = lantai;
            kendaraan = k;
            break;
        }
    }

    if (lantaiAsal == null || kendaraan == null) {
        System.out.println("Kendaraan dengan nomor plat " + nomorPlat + " tidak ditemukan!");
        return false;
    }

    System.out.println("Kendaraan ditemukan di lantai " + lantaiAsal.getNomorLantai());
    System.out.print("Masukkan nomor lantai tujuan (1-" + daftarLantai.size() + "): ");

    int nomorLantaiTujuan;
    try {
        nomorLantaiTujuan = Integer.parseInt(scanner.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Input tidak valid!");
        return false;
    }

    if (nomorLantaiTujuan < 1 || nomorLantaiTujuan > daftarLantai.size()) {
        System.out.println("Nomor lantai tidak valid!");
        return false;
    }

    if (nomorLantaiTujuan == lantaiAsal.getNomorLantai()) {
```

```
            System.out.println("Kendaraan sudah berada di lantai " + nomorLantaiTujuan +
"!");
            return false;
        }

        Lantai lantaiTujuan = daftarLantai.get(nomorLantaiTujuan - 1);

        if (lantaiTujuan.getSisaKapasitasRoda() >= kendaraan.getJumlahRoda()) {
            kendaraan = lantaiAsal.keluarkanKendaraan(nomorPlat);
            lantaiTujuan.tambahKendaraan(kendaraan);
            System.out.println(kendaraan.getJenisKendaraan() + " dengan nomor plat " +
nomorPlat +
                    " berhasil dipindahkan dari lantai " + lantaiAsal.getNomorLantai() +
                    " ke lantai " + lantaiTujuan.getNomorLantai());
            return true;
        } else {
            System.out.println("Lantai " + nomorLantaiTujuan + " tidak cukup ruang untuk
kendaraan ini!");
            return false;
        }
    }

    public boolean keluarkanKendaraan(Scanner scanner) {
        System.out.println("\n===== KELUARKAN KENDARAAN =====");
        System.out.print("Masukkan nomor plat kendaraan yang akan dikeluarkan: ");
        String nomorPlat = scanner.nextLine();

        Lantai lantaiParkir = null;
        Kendaraan kendaraan = null;

        for (Lantai lantai : daftarLantai) {
```

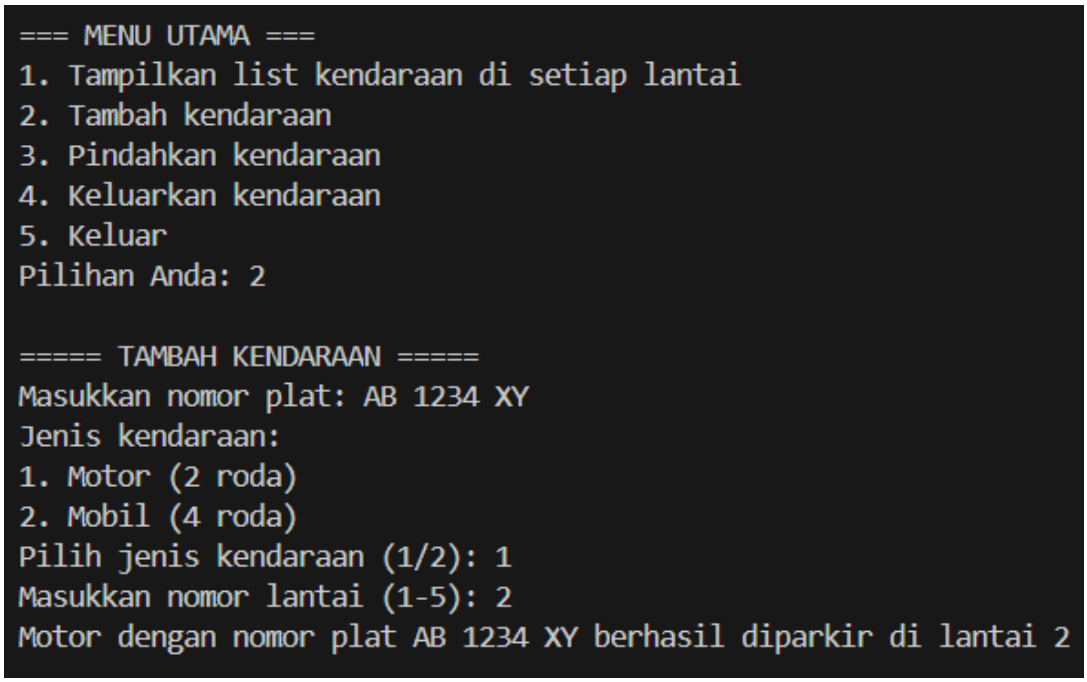
```
            Kendaraan k = lantai.cariKendaraan(nomorPlat);
            if (k != null) {
                lantaiParkir = lantai;
                kendaraan = k;
                break;
            }
        }

        if (lantaiParkir == null || kendaraan == null) {
            System.out.println("Kendaraan dengan nomor plat " + nomorPlat + " tidak ditemukan!");
            return false;
        }

        kendaraan = lantaiParkir.keluarkanKendaraan(nomorPlat);

        LocalDateTime waktuKeluar = LocalDateTime.now();
        long durasiJam = ChronoUnit.HOURS.between(kendaraan.getWaktuMasuk(), waktuKeluar);
        if (durasiJam < 1) durasiJam = 1;

        int biaya;
        if (kendaraan.getJumlahRoda() == 2) {
            biaya = (int) (durasiJam * BIAYA_PARKIR_PER_JAM_MOTOR);
        } else {
            biaya = (int) (durasiJam * BIAYA_PARKIR_PER_JAM_MOBIL);
        }

        System.out.println(kendaraan.getJenisKendaraan() + " dengan nomor plat " + nomorPlat +
                           " berhasil dikeluarkan dari lantai " + lantaiParkir.getNomorLantai());
```

```
        System.out.println("Durasi parkir: " + durasiJam + " jam");
        System.out.println("Biaya parkir: Rp " + formatCurrency(biaya));
        return true;
    }

    private String formatCurrency(int amount) {
        StringBuilder formatted = new StringBuilder();
        String amountStr = String.valueOf(amount);
        int length = amountStr.length();

        for (int i = 0; i < length; i++) {
            formatted.append(amountStr.charAt(i));
            if ((length - i - 1) % 3 == 0 && i < length - 1) {
                formatted.append('.');
            }
        }

        return formatted.toString();
    }
}

public class SistemManajemenParkir {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        SistemParkir sistemParkir = new SistemParkir(5);
        int pilihan = 0;

        System.out.println("=== SISTEM MANAJEMEN PARKIR ===");
        System.out.println("Kapasitas: 5 lantai, masing-masing maksimal 12 roda");

        do {
```

```
            System.out.println("\n=== MENU UTAMA ===");
            System.out.println("1. Tampilkan list kendaraan di setiap lantai");
            System.out.println("2. Tambah kendaraan");
            System.out.println("3. Pindahkan kendaraan");
            System.out.println("4. Keluarkan kendaraan");
            System.out.println("5. Keluar");
            System.out.print("Pilihan Anda: ");

            try {
                pilihan = Integer.parseInt(scanner.nextLine());

                switch (pilihan) {
                    case 1:
                        sistemParkir.tampilkanDaftarKendaraan();
                        break;
                    case 2:
                        sistemParkir.tambahKendaraan(scanner);
                        break;
                    case 3:
                        sistemParkir.pindahkanKendaraan(scanner);
                        break;
                    case 4:
                        sistemParkir.keluarkanKendaraan(scanner);
                        break;
                    case 5:
                        System.out.println("Terima kasih telah menggunakan Sistem Manajemen
Parkir!");
                        break;
                    default:
                        System.out.println("Pilihan tidak valid!");
                }
```

```
            } catch (NumberFormatException e) {
                System.out.println("Input tidak valid!");
            }

        } while (pilihan != 5);

        scanner.close();
    }
}
```

2. Hasil run
    a. Menambahkan Kendaraan



    b. Menampilkan Kendaraan

```
=== MENU UTAMA ===
1. Tampilkan list kendaraan di setiap lantai
2. Tambah kendaraan
3. Pindahkan kendaraan
4. Keluarkan kendaraan
5. Keluar
Pilihan Anda: 1

===== DAFTAR KENDARAAN PARKIR =====
Lantai 1:
Jumlah roda saat ini: 4 | Sisa kapasitas: 8 roda
1. Mobil - Plat: B 312 UK | Roda: 4 | Masuk: 18-03-2025 16:54:24

Lantai 2:
Jumlah roda saat ini: 2 | Sisa kapasitas: 10 roda
1. Motor - Plat: AB 1234 XY | Roda: 2 | Masuk: 18-03-2025 16:52:21

Lantai 3:
Jumlah roda saat ini: 0 | Sisa kapasitas: 12 roda
- Tidak ada kendaraan

Lantai 4:
Jumlah roda saat ini: 0 | Sisa kapasitas: 12 roda
- Tidak ada kendaraan

Lantai 5:
Jumlah roda saat ini: 4 | Sisa kapasitas: 8 roda
1. Mobil - Plat: N 1 GZ | Roda: 4 | Masuk: 18-03-2025 16:54:57
```

c. Memindah Kendaraan

```
=== MENU UTAMA ===
1. Tampilkan list kendaraan di setiap lantai
2. Tambah kendaraan
3. Pindahkan kendaraan
4. Keluarkan kendaraan
5. Keluar
Pilihan Anda: 3

===== PINDAHKAN KENDARAAN =====
Masukkan nomor plat kendaraan yang akan dipindahkan: B 312 UK
Kendaraan ditemukan di lantai 1
Masukkan nomor lantai tujuan (1-5): 2
Mobil dengan nomor plat B 312 UK berhasil dipindahkan dari lantai 1 ke lantai 2
```

d. Mengeluarkan Kendaraan

```
===== KELUARKAN KENDARAAN =====
Masukkan nomor plat kendaraan yang akan dikeluarkan: N 1 GZ
Mobil dengan nomor plat N 1 GZ berhasil dikeluarkan dari lantai 5
Durasi parkir: 1 jam
Biaya parkir: Rp 5.000
```

e. Keluar Program

```
=== MENU UTAMA ===
1. Tampilkan list kendaraan di setiap lantai
2. Tambah kendaraan
3. Pindahkan kendaraan
4. Keluarkan kendaraan
5. Keluar
Pilihan Anda: 5
Terima kasih telah menggunakan Sistem Manajemen Parkir!
```

3. Penjelasan

a. Struktur kode

Kode ini terdiri dari beberapa kelas utama:

- Kendaraan → Merepresentasikan objek kendaraan yang diparkir.
- Lantai → Merepresentasikan satu lantai parkir yang memiliki kapasitas tertentu.
- SistemParkir → Merepresentasikan sistem parkir dengan beberapa lantai dan fungsionalitas seperti parkir, pindah kendaraan, dan keluarkan kendaraan.
- SistemManajemenParkir (Main Class) → Kelas utama yang menjalankan program dan berinteraksi dengan pengguna melalui menu berbasis teks.

b. Class Kendaraan

```
class Kendaraan {
   private String nomorPlat;
   private int jumlahRoda;
   private LocalDateTime waktuMasuk;
}
```

1. Atribut:
   - nomorPlat: Nomor kendaraan.
   - jumlahRoda: Jumlah roda kendaraan (2 = motor, 4 = mobil).

- waktuMasuk: Waktu kendaraan masuk ke parkiran.

2. Fungsi utama:
   - Konstruktor Kendaraan(String nomorPlat, int jumlahRoda) → Mengatur nomor plat dan jumlah roda kendaraan, serta menyimpan waktu masuk kendaraan menggunakan LocalDateTime.now().
   - getJenisKendaraan() → Mengembalikan jenis kendaraan berdasarkan jumlah roda (motor/mobil).
   - toString() → Mengembalikan informasi kendaraan dalam bentuk string.

c. Class Lantai

```
class Lantai {
   private int nomorLantai;
   private List<Kendaraan> daftarKendaraan;
   private final int KAPASITAS_RODA = 12;
}
```

1. Atribut:
   - nomorLantai: Nomor lantai parkir.
   - daftarKendaraan: List kendaraan yang terparkir di lantai ini.
   - KAPASITAS_RODA: Maksimal jumlah roda yang dapat diparkir di lantai (12 roda).
2. Fungsi utama:
   - tambahKendaraan(Kendaraan kendaraan) → Menambahkan kendaraan ke lantai jika masih ada kapasitas.
   - keluarkanKendaraan(String nomorPlat) → Mengeluarkan kendaraan berdasarkan nomor plat.
   - cariKendaraan(String nomorPlat) → Mencari kendaraan berdasarkan nomor plat.
   - getSisaKapasitasRoda() → Menghitung sisa kapasitas roda di lantai ini.

d. Class SistemParkir

```
class SistemParkir {
   private List<Lantai> daftarLantai;
   private final int BIAYA_PARKIR_PER_JAM_MOTOR = 2000;
```

```
    private final int BIAYA_PARKIR_PER_JAM_MOBIL = 5000;
}
```

1. Atribut:
   - daftarLantai: List lantai dalam sistem parkir.
   - BIAYA_PARKIR_PER_JAM_MOTOR: Biaya parkir motor per jam (Rp 2.000).
   - BIAYA_PARKIR_PER_JAM_MOBIL: Biaya parkir mobil per jam (Rp 5.000).
2. Fungsi utama:
   - tambahKendaraan(Scanner scanner) → Memasukkan kendaraan ke parkiran berdasarkan input pengguna.
   - pindahkanKendaraan(Scanner scanner) → Memindahkan kendaraan ke lantai lain.
   - keluarkanKendaraan(Scanner scanner) → Mengeluarkan kendaraan dan menghitung biaya parkir.
   - tampilkanDaftarKendaraan() → Menampilkan semua kendaraan yang sedang parkir.

e. Class SistemManajemenParkir

```
public class SistemManajemenParkir {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        SistemParkir sistemParkir = new SistemParkir(5);
        int pilihan = 0;
```

1. Program utama:
   - Menggunakan menu berbasis teks untuk mengelola parkiran.
   - Memanggil fungsi tambahKendaraan(), pindahkanKendaraan(), keluarkanKendaraan(), dan tampilkanDaftarKendaraan().
   - Menggunakan perulangan do-while untuk terus menampilkan menu hingga pengguna memilih keluar.

f. Alur kerja program

1. Menampilkan Menu → Pengguna memilih aksi yang ingin dilakukan.

2. Menambah Kendaraan → Kendaraan ditambahkan ke lantai yang masih memiliki kapasitas cukup.
3. Memindahkan Kendaraan → Kendaraan bisa dipindah ke lantai lain jika ada tempat.
4. Mengeluarkan Kendaraan → Kendaraan dihapus dari daftar parkiran, dan biaya parkir dihitung berdasarkan durasi.
5. Menampilkan Data Kendaraan → Menampilkan daftar kendaraan yang saat ini parkir di setiap lantai.