LAPORAN PRAKTIKUM

JAVA OBJECT ORIENTED PROGRAMMING

"METHODS, REFERENSE PADA OBJECT, DAN ACCESS MODIFIER"

Oleh:

Benony Gabriel

105222002

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN ILMU KOMPUTER

UNIVERSITAS PERTAMINA

2024

1. Pendahuluan

1.1 Studi Kasus

Anda diminta untuk merancang sebuah sistem sederhana untuk reservasi hotel. Sistem ini terdiri dari beberapa class, yaitu Room, Hotel, dan Reservation. Berikut adalah spesifikasi untuk masing-masing class:

- Class Room

  Variabel:

  a. int roomNumber
  b. String roomType
  c. boolean occupied
  d. double rate

  Constructor:

  a. Constructor default yang menginisialisasi roomNumber dengan 0, roomType dengan "Standard", occupied dengan false, dan rate dengan 100.0.
  b. Constructor yang menerima roomNumber dan roomType, dan menginisialisasi occupied dengan false serta rate berdasarkan roomType. (Misal: "Standard" = 100.0, "Deluxe" = 200.0)
  c. Constructor yang menerima semua parameter untuk variabelnya.

- Class Hotel

Variabel:

a. String name
b. Room[] rooms
c. Constructor:

   Constructor yang menerima name dan int numberOfRooms. Inisialisasi array rooms dengan jumlah numberOfRooms. Gunakan sebuah loop untuk mengisi array rooms dengan objek Room, dimana roomNumber dimulai dari 1 hingga numberOfRooms dan semua kamar bertipe "Standard".

   Constructor yang menerima name, int numberOfStandardRooms, dan int numberOfDeluxeRooms. Inisialisasi array rooms dengan jumlah total kamar. Gunakan sebuah loop untuk mengisi rooms dengan objek Room, dimana setengah pertama adalah "Standard" dan setengah kedua adalah "Deluxe".

- Class Reservation

  Variabel:

  a. Room reservedRoom
  b. String guestName
  c. int numberOfNights
  d. double totalCharge

Constructor:

Constructor yang menerima Room reservedRoom, String guestName, dan int numberOfNights. totalCharge dihitung berdasarkan rate dari reservedRoom dikalikan dengan numberOfNights.

1.2 Tugas

a. Implementasikan semua class dan constructor sesuai dengan spesifikasi di atas. Pastikan Anda menerapkan prinsip encapsulation dengan membuat variabel instance private dan menyediakan getter (dan setter jika diperlukan).
b. Untuk setiap constructor, tambahkan kode untuk mencetak pesan yang menunjukkan constructor mana yang sedang dipanggil dan parameter apa saja yang diberikan. Contoh: "Constructor Room with roomNumber and roomType is called".
c. Buat class HotelReservationSystem dengan method main untuk demonstrasi:
d. Buat sebuah objek Hotel dengan kedua constructor yang tersedia.
e. Buat sebuah objek Reservation untuk beberapa kamar.
f. Cetak informasi mengenai reservasi yang telah dibuat.

2. Variabel

Berikut ini adalah penjelasan singkat mengenai variabel-variabel yang dibuat di setiap kelas:

2.1 Class Room

No	Nama Variabel	Tipe data	Fungsi
1	roomNumber	int	Menyimpan nomor kamar.
2	roomType	String	Menyimpan tipe kamar ("Standard" atau "Deluxe").
3	occupied	boolean	Menyimpan status ketersediaan kamar (true jika terisi, false jika kosong).
4	rate	double	Menyimpan tarif kamar per malam.

2.2 Class Hotel

No	Nama Variabel	Tipe data	Fungsi
1	name	String	Menyimpan nama hotel.
2	rooms	Room	Menyimpan array dari objek Room, mewakili kamar-kamar yang tersedia di hotel.

2.3 Class Reservation

No	Nama Variabel	Tipe data	Fungsi
1	reservedRoom	Room	Menyimpan objek Room yang telah dipesan.
2	guestName	String	Menyimpan nama tamu yang memesan kamar.
3	numberOfNights	int	Menyimpan jumlah malam yang dipesan.
4	totalCharge	double	Menyimpan total biaya pemesanan kamar.

Setiap variabel ini memainkan peran penting dalam kelasnya masing-masing dan digunakan untuk menyimpan informasi yang berkaitan dengan objek yang dibuat dari kelas tersebut. Dengan menggunakan variabel-variabel ini, kita dapat dengan mudah melacak dan mengelola informasi yang terkait dengan kamar hotel dan reservasi.

3. Constructor dan Method

3.1 Class Room

No	Nama Metode	Jenis	Fungsi
1	Room(int, String)	Constructor	Membuat objek Room dengan nomor dan tipe kamar tertentu.
2	Room()	Constructor	Membuat objek Room dengan nilai default.
3	Room(int, String, boolean, double)	Constructor	Membuat objek Room dengan semua atribut yang ditentukan.
4	getRoomNumber()	Function	Mengembalikan nomor kamar.
5	setRoomNumber(int)	Procedural	Mengatur nomor kamar.
6	getRoomType()	Function	Mengembalikan tipe kamar.
7	setRoomType(String)	Procedural	Mengatur tipe kamar.
8	isOccupied()	Function	Mengembalikan status kamar (terisi atau tidak).
9	setOccupied(boolean)	Procedural	Mengatur status kamar.
10	getRate()	Function	Mengembalikan tarif kamar.
11	setRate(double)	Procedural	Mengatur tarif kamar.

3.2 Class Hotel

No	Nama Metode	Jenis	Fungsi
1	Hotel(String, int)	Constructor	Membuat objek Hotel dengan nama dan jumlah kamar tertentu.

2	Hotel(String, int, int)	Constructor	Membuat objek Hotel dengan jumlah kamar standar dan mewah.
3	bookRoom(int)	Procedural	Memesan kamar dengan nomor tertentu.
4	cancelBooking(int)	Procedural	Membatalkan pemesanan kamar dengan nomor tertentu.
5	checkRoomAvailability(int)	Function	Memeriksa ketersediaan kamar dengan nomor tertentu.
6	displayRoomDetails(int)	Procedural	Menampilkan detail kamar dengan nomor tertentu.

3.3 Class Reservation

No	Nama Metode	Jenis	Fungsi
1	Reservation(Room, String, int)	Constructor	Membuat objek Reservasi dengan kamar, nama tamu, dan malam menginap.
2	calculateTotalCharge()	Function	Menghitung total biaya pemesanan kamar.
3	displayReservationDetails()	Procedural	Menampilkan detail pemesanan kamar.

5. Dokumentasi dan Pembahasan Code

5.1 Class Room

Room.java

```java
public class Room {
    private int roomNumber;
    private String roomType;
    private boolean occupied;
    private double rate;

    Room(){
        this.roomNumber = 0;
        this.roomType = "Standard";
        this.occupied = false;
        this.rate = 100.0d;

        System.out.println("Constructor Room default is called");
    }

    Room(int roomNumber, String roomType){
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.occupied = false;

        if(this.roomType == "Standard"){
            this.rate = 100.0d;
        }else if(this.roomType == "Deluxe"){
            this.rate = 200.0d;
        }else{
            System.out.println("Tipe kamar tidak tersedia!");
        }

        System.out.println("Constructor Room with roomNumber and roomType is called");
    }

    Room(int roomNumber, String roomType, boolean occupied, double rate){
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.occupied = occupied;
        this.rate = rate;

        System.out.println("Constructor Room with all parameters is called");
    }

    // Buat fungsi setter dan getter
    public int getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;
    }

    public String getRoomType() {
        return roomType;
    }

    public void setRoomType(String roomType) {
        this.roomType = roomType;
    }

    public boolean isOccupied() {
        return occupied;
    }

    public void setOccupied(boolean occupied) {
        this.occupied = occupied;
    }

    public double getRate() {
        return rate;
    }

    public void setRate(double rate) {
        this.rate = rate;
    }

}
```

Kode di atas adalah implementasi dari kelas `Room` yang digunakan untuk merepresentasikan kamar hotel. Berikut adalah penjelasan singkat mengenai kode tersebut:

1. Atribut:
   - `roomNumber` (int): Nomor kamar.
   - `roomType` (String): Tipe kamar.
   - `occupied` (boolean): Status ketersediaan kamar.
   - `rate` (double): Tarif kamar.

2. Constructor:
   - `Room()`: Constructor default yang menginisialisasi atribut dengan nilai default.
   - `Room(int roomNumber, String roomType)`: Constructor yang menerima nomor kamar dan tipe kamar, dan menginisialisasi atribut sesuai dengan parameter yang diberikan.
   - `Room(int roomNumber, String roomType, boolean occupied, double rate)`: Constructor yang menerima semua parameter untuk atribut, dan menginisialisasi atribut sesuai dengan parameter yang diberikan.

3. Getter dan Setter:
   - Metode getter digunakan untuk mengembalikan nilai atribut.
   - Metode setter digunakan untuk mengatur nilai atribut.

4. Pesan Output:
   - Pada setiap constructor, terdapat perintah untuk mencetak pesan yang menunjukkan constructor mana yang dipanggil dan parameter apa saja yang diberikan. Pesan ini digunakan untuk tujuan debugging dan pemahaman saat pengembangan aplikasi.

Kelas `Room` ini dirancang dengan prinsip encapsulation, di mana atributnya memiliki akses modifier private sehingga hanya dapat diakses melalui metode getter dan setter. Hal ini memungkinkan untuk melindungi data dari modifikasi yang tidak sah dan memastikan konsistensi dan keamanan data. Selain itu, konstruktor yang disediakan memungkinkan objek `Room` untuk dibuat dengan berbagai konfigurasi sesuai kebutuhan aplikasi.

5.2 Class Hotel

Hotel.java

```java
public class Hotel {
    private String name;
    private Room[] rooms;

    public Hotel(String name, int numberOfRooms) {
        this.name = name;
        this.rooms = new Room[numberOfRooms];
        for (int i = 0; i < numberOfRooms; i++) {
            rooms[i] = new Room(i + 1, "Standard");
        }
        System.out.println("Constructor Hotel with name and numberOfRooms is called");
    }

    public Hotel(String name, int numberOfStandardRooms, int numberOfDeluxeRooms) {
        this.name = name;
        int totalRooms = numberOfStandardRooms + numberOfDeluxeRooms;
        this.rooms = new Room[totalRooms];
        for (int i = 0; i < totalRooms; i++) {
            if (i < numberOfStandardRooms) {
                rooms[i] = new Room(i + 1, "Standard");
            } else {
                rooms[i] = new Room(i + 1, "Deluxe");
            }
        }
        System.out.println("Constructor Hotel with name, numberOfStandardRooms, and numberOfDeluxeRooms is called");
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Room[] getRooms() {
        return rooms;
    }

    public void setRooms(Room[] rooms) {
        this.rooms = rooms;
    }
}
```

Kode di atas adalah implementasi dari kelas `Hotel` yang digunakan untuk merepresentasikan sebuah hotel. Berikut adalah penjelasan singkat mengenai kode tersebut:

1. Atribut:
   - `name` (String): Nama hotel.
   - `rooms` (Room[]): Array of objects dari kelas `Room` yang merepresentasikan kamar-kamar dalam hotel.

2. Constructor:
   - `Hotel(String name, int numberOfRooms)`: Constructor yang menerima nama hotel dan jumlah kamar. Konstruktor ini membuat array `rooms` dengan panjang yang sesuai dan menginisialisasi setiap elemen array dengan objek `Room` yang mewakili kamar-kamar standar.
   - `Hotel(String name, int numberOfStandardRooms, int numberOfDeluxeRooms)`: Constructor yang menerima nama hotel, jumlah kamar standar, dan jumlah kamar

mewah. Konstruktor ini membuat array `rooms` dengan panjang yang sesuai, di mana setengah pertama elemen array diisi dengan objek `Room` yang mewakili kamar standar dan setengah kedua diisi dengan objek `Room` yang mewakili kamar mewah.

3. Getter dan Setter:
   - Metode getter digunakan untuk mengembalikan nilai atribut.
   - Metode setter digunakan untuk mengatur nilai atribut.

4. Pesan Output:
   - Pada setiap constructor, terdapat perintah untuk mencetak pesan yang menunjukkan constructor mana yang dipanggil dan parameter apa saja yang diberikan. Pesan ini digunakan untuk tujuan debugging dan pemahaman saat pengembangan aplikasi.

Kelas `Hotel` ini juga dirancang dengan prinsip encapsulation, di mana atributnya memiliki akses modifier private sehingga hanya dapat diakses melalui metode getter dan setter. Hal ini memungkinkan untuk melindungi data dari modifikasi yang tidak sah dan memastikan konsistensi dan keamanan data. Selain itu, konstruktor yang disediakan memungkinkan objek `Hotel` untuk dibuat dengan berbagai konfigurasi sesuai kebutuhan aplikasi.

5.3 Class Resevation

Reservation.java

```java
public class Reservation {
    private Room reservedRoom;
    private String guestName;
    private int numberOfNights;
    private double totalCharge;

    public Reservation(Room reservedRoom, String guestName, int numberOfNights) {
        this.reservedRoom = reservedRoom;
        this.guestName = guestName;
        this.numberOfNights = numberOfNights;
        calculateTotalCharge();
        System.out.println("Constructor Reservation is called");
    }

    private void calculateTotalCharge() {
        this.totalCharge = reservedRoom.getRate() * numberOfNights;
    }

    public Room getReservedRoom() {
        return reservedRoom;
    }

    public void setReservedRoom(Room reservedRoom) {
        this.reservedRoom = reservedRoom;
    }

    public String getGuestName() {
        return guestName;
    }

    public void setGuestName(String guestName) {
        this.guestName = guestName;
    }

    public int getNumberOfNights() {
        return numberOfNights;
    }

    public void setNumberOfNights(int numberOfNights) {
        this.numberOfNights = numberOfNights;
        calculateTotalCharge();
    }

    public double getTotalCharge() {
        return totalCharge;
    }
}
```

Kode di atas adalah implementasi dari kelas `Reservation`, yang digunakan untuk merepresentasikan sebuah reservasi hotel. Berikut adalah penjelasan singkat mengenai kode tersebut:

1. Atribut:
   - `reservedRoom` (Room): Objek `Room` yang merepresentasikan kamar yang dipesan.
   - `guestName` (String): Nama tamu yang melakukan reservasi.
   - `numberOfNights` (int): Jumlah malam menginap.
   - `totalCharge` (double): Total biaya reservasi.

2. Constructor:
   - `Reservation(Room reservedRoom, String guestName, int numberOfNights)`: Constructor yang menerima objek `Room` yang dipesan, nama tamu, dan jumlah malam menginap. Konstruktor ini menginisialisasi atribut-atribut sesuai dengan nilai yang diberikan dan kemudian memanggil metode `calculateTotalCharge()` untuk menghitung total biaya reservasi.

3. Metode `calculateTotalCharge()`:
   - Metode ini adalah metode private yang digunakan untuk menghitung total biaya reservasi berdasarkan rate dari kamar yang dipesan dan jumlah malam menginap.

4. Getter dan Setter:
   - Metode getter digunakan untuk mengembalikan nilai atribut.
   - Metode setter digunakan untuk mengatur nilai atribut. Metode `setNumberOfNights()` juga memanggil metode `calculateTotalCharge()` setelah mengatur nilai `numberOfNights` agar total biaya reservasi dapat diperbarui.

Kelas `Reservation` ini juga dirancang dengan prinsip encapsulation, di mana atributnya memiliki akses modifier private sehingga hanya dapat diakses melalui metode getter dan setter. Hal ini memungkinkan untuk melindungi data dari modifikasi yang tidak sah dan memastikan konsistensi dan keamanan data. Selain itu, konstruktor yang disediakan memungkinkan objek `Reservation` untuk dibuat dengan parameter yang sesuai untuk melakukan reservasi dengan berbagai konfigurasi.

5.4 Class ReservationSystem

HotelReservationSystem.java

```java
// File: HotelReservationSystem.java
public class HotelReservationSystem {
    public static void main(String[] args) {
        // Buat hotel dengan 5 ruangan standar
        Hotel hotel1 = new Hotel("Grand Hotel", 5);

        // Buat hotel dengan 3 ruangan standar dan 2 ruangan deluxe
        Hotel hotel2 = new Hotel("Luxury Resort", 3, 2);

        // Buat sebuah resevasi utk ruangan 1 di hotel1 untuk 3 malam
        Reservation reservation1 = new Reservation(hotel1.getRooms()[0], "Benony Gabriel", 3);

        // Buat sebuah reservasi untuk 2 ruangan di hotel untuk 5 malam
        Reservation reservation2 = new Reservation(hotel2.getRooms()[1], "Randy Orton", 5);

        // Menampilkan informasi tentang reservasi
        System.out.println("Reservation 1:");
        System.out.println("Guest Name: " + reservation1.getGuestName());
        System.out.println("Room Number: " + reservation1.getReservedRoom().getRoomNumber());
        System.out.println("Number of Nights: " + reservation1.getNumberOfNights());
        System.out.println("Total Charge: Rp." + reservation1.getTotalCharge());

        System.out.println();

        System.out.println("Reservation 2:");
        System.out.println("Guest Name: " + reservation2.getGuestName());
        System.out.println("Room Number: " + reservation2.getReservedRoom().getRoomNumber());
        System.out.println("Number of Nights: " + reservation2.getNumberOfNights());
        System.out.println("Total Charge: Rp." + reservation2.getTotalCharge());
    }
}
```

6. Kesimpulan

Dari keseluruhan implementasi yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. Penerapan Konsep OOP:
   - Implementasi studi kasus reservasi hotel menggunakan konsep Object-Oriented Programming (OOP) dengan baik.
   - Setiap entitas seperti `Room`, `Hotel`, dan `Reservation` direpresentasikan sebagai kelas-kelas dengan properti-properti yang sesuai.

2. Penggunaan Konstruktor:
   - Konstruktor digunakan untuk menginisialisasi objek-objek dengan nilai-nilai awal yang sesuai.
   - Penerapan konstruktor memungkinkan pembuatan objek dengan berbagai opsi dan parameter yang fleksibel.

3. Encapsulation:

- Encapsulation diterapkan dengan baik dengan membuat properti-properti kelas sebagai private dan menyediakan metode-metode getter dan setter untuk mengakses dan memanipulasi properti tersebut.

4. Penggunaan Array:
   - Array digunakan untuk menyimpan objek-objek `Room` dalam kelas `Hotel`.
   - Penggunaan array memudahkan dalam mengelola dan mengakses sejumlah objek yang memiliki tipe data yang sama.

5. Penanganan Informasi:
   - Informasi tentang reservasi hotel seperti nama tamu, nomor kamar, jumlah malam menginap, dan total biaya reservasi ditangani dengan baik dan akurat.
   - Metode-metode yang ada memungkinkan untuk mengambil dan mengatur informasi dengan mudah.

6. Prinsip Keterbacaan dan Keterorganisasian Kode:
   - Kode ditulis dengan jelas dan terstruktur sehingga mudah dipahami dan diikuti.
   - Penamaan variabel, fungsi, dan kelas yang konsisten membantu dalam memahami tujuan dan fungsi masing-masing bagian kode.

7. Penggunaan Main Class:
   - Kode main yang ada di kelas `HotelReservationSystem` berfungsi sebagai entry point dan memudahkan dalam menjalankan program serta melakukan operasi-operasi yang diperlukan.

Secara keseluruhan, implementasi studi kasus reservasi hotel telah berhasil dilakukan dengan baik sesuai dengan spesifikasi yang telah diberikan. Dengan menggunakan konsep OOP, penggunaan konstruktor, encapsulation, dan keterorganisasian kode yang baik, sistem reservasi hotel dapat dijalankan dan dikelola dengan efisien dan efektif.