

LAPORAN PRAKTIKUM WEEK 6
PEMROGRAMAN BERORIENTASI OBJEK
“ARRAY LIST DAN PACKAGE”



Oleh
Benony Gabriel
105222002

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS SAINS DAN ILMU KOMPUTER
UNIVERSITAS PERTAMINA
2024

1. Pendahuluan

1.1 Array List

ArrayList adalah kelas dalam Java yang terdapat dalam paket `java.util`. Ini adalah implementasi dari struktur data dinamis yang dapat menampung sejumlah besar elemen. ArrayList memungkinkan penyimpanan dan manipulasi elemen-elemen dengan cara yang lebih fleksibel daripada array biasa.

Berikut adalah beberapa karakteristik dan keuntungan dari ArrayList:

- **Ukuran Dinamis:** ArrayList dapat menyesuaikan ukurannya secara otomatis saat elemen-elemen ditambahkan atau dihapus. Anda tidak perlu menentukan ukuran awal saat membuat ArrayList.
- **Akses Cepat:** Anda dapat mengakses elemen-elemen ArrayList menggunakan indeks. Ini memungkinkan pencarian, penyisipan, dan penghapusan elemen dengan cepat.
- **Penambahan dan Penghapusan Efisien:** Menambah atau menghapus elemen di ujung ArrayList (awal atau akhir) memiliki kompleksitas waktu $O(1)$. Namun, untuk operasi penambahan atau penghapusan di tengah ArrayList, kompleksitas waktu adalah $O(n)$, di mana n adalah jumlah elemen yang perlu dipindahkan.
- **Fleksibilitas:** Anda dapat menyimpan elemen-elemen dari tipe data apa pun dalam ArrayList, termasuk objek dan tipe primitif.
- **Iterator:** ArrayList menyediakan iterator yang memungkinkan Anda untuk melakukan iterasi atau traverse melalui elemen-elemen dalam urutan tertentu.

1.2 Package

Dalam Java, package adalah cara untuk mengorganisir kelas-kelas ke dalam kelompok yang terkait. Package membantu dalam manajemen kode, memperjelas struktur proyek, dan menghindari konflik nama antar kelas.

Berikut adalah beberapa poin penting tentang package dalam Java:

- **Organisasi Kode:** Package memungkinkan Anda untuk mengorganisir kelas-kelas Anda ke dalam kelompok yang bermakna. Misalnya, Anda dapat memiliki package untuk bagian-bagian tertentu dari aplikasi Anda, seperti package untuk model data, package untuk logika bisnis, dan package untuk antarmuka pengguna.
- **Nama Unik:** Nama kelas dalam Java harus unik di seluruh proyek. Dengan menggunakan package, Anda dapat memiliki kelas dengan nama yang sama dalam package yang berbeda tanpa menyebabkan konflik.
- **Visibility Control:** Package juga digunakan untuk mengontrol aksesibilitas kelas dan anggota kelas (variabel, metode, dll.). Anda dapat menggunakan access modifiers seperti `public`, `protected`, atau

default (tidak ada access modifier) untuk mengatur tingkat aksesibilitas kelas dan anggota kelas.

- Import Statement: Untuk menggunakan kelas atau anggota kelas dari package lain, Anda perlu mengimpornya ke dalam file Java Anda menggunakan pernyataan import. Ini memberi tahu kompiler di mana untuk mencari kelas yang diimpor saat digunakan dalam kode.
- Paket Standar: Java memiliki sejumlah paket standar yang menyediakan kelas-kelas untuk fungsionalitas yang umum digunakan, seperti java.util untuk struktur data, java.io untuk operasi input/output, dan banyak lagi. Anda dapat menggunakan kelas-kelas dari paket ini tanpa perlu membuatnya sendiri.
- Package Naming Convention: Konvensi penamaan package di Java adalah menggunakan nama domain Anda terbalik sebagai awalan. Misalnya, jika nama domain Anda adalah example.com, Anda dapat menggunakan com.example sebagai awalan untuk package Anda.

Dengan menggunakan package, kita dapat membuat struktur proyek yang terorganisir dengan baik dan memfasilitasi pengembangan dan pemeliharaan kode yang lebih mudah.

1.3 Studi Kasus : Sistem Pemesanan Makanan Online

Anda diminta untuk mengembangkan bagian dari sistem pemesanan makanan online. Sistem ini terdiri dari beberapa modul yang dibagi ke dalam package yang berbeda, dengan kebutuhan untuk mengontrol akses ke data secara ketat menggunakan access modifiers.

1.3.1 Struktur Package dan Kelas

1.3.1.1 Package menu:

- Kelas MenuItem: Menyimpan detail item menu seperti id (String), nama makanan (String), dan harga (double). Atribut harus diatur agar tidak dapat diakses langsung dari luar kelas, tapi bisa diakses melalui method di dalam package yang sama.
- Kelas MenuDatabase: Menggunakan ArrayList untuk menyimpan daftar MenuItem. Kelas ini harus menyediakan method untuk menambahkan item baru ke dalam menu, menghapus item berdasarkan id, dan mengembalikan salinan daftar menu item saat ini yang tidak memungkinkan modifikasi langsung pada daftar asli.

1.3.1.2 Package order:

- Kelas Order: Mencatat detail pesanan, termasuk id pesanan (String) dan item-item yang dipesan (menggunakan ArrayList dari MenuItem). Pastikan item dalam pesanan tidak dapat

diakses atau dimodifikasi langsung dari luar kelas, tapi dapat ditambahkan melalui method di kelas ini.

- Kelas OrderManager: Mengelola daftar pesanan menggunakan ArrayList. Kelas ini harus menyediakan fungsi untuk membuat pesanan baru, menghapus pesanan berdasarkan id, dan mengembalikan informasi tentang pesanan.

1.3.1.3 Package util:

- Kelas IDGenerator: Menghasilkan ID unik untuk item menu dan pesanan. Fungsi ini harus dapat diakses oleh kelas di package menu dan order.

1.3.2 Persyaratan dan Batasan:

- Gunakan access modifiers (private, package-private, public) untuk memastikan bahwa data diakses dan dimodifikasi hanya melalui method yang diperbolehkan.
- Dilarang menggunakan inheritance dan polymorphism dalam solusi.
- Demonstrasi penggunaan efektif ArrayList untuk menyimpan dan manipulasi objek.
- Pastikan bahwa daftar item menu dan pesanan dapat diakses secara aman tanpa memungkinkan modifikasi eksternal yang tidak sah pada struktur data internal.

1.3.3 Poin Tambahan:

- Implementasikan metode dalam MenuDatabase dan OrderManager untuk mencari item atau pesanan berdasarkan nama atau id.
- Tambahkan validasi dalam method untuk menangani kasus seperti duplikasi id saat menambahkan item baru ke menu atau membuat pesanan baru.

2. Variabel dan Tipe Data

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

2.1 Package Menu

Didalam package Menu terdapat dua buah kelas yaitu kelas MenuItem dan MenuDatabase. Berikut ini adalah deskripsi singkat mengenai variabel-variabel yang dibuat di kedua kelas tersebut.

2.1.1 Class MenuItem

Nama Variabel	Tipe Data	Keterangan
Id	String	Untuk menomori setiap item menu
Name	String	Untuk menyimpan nama menu
Price	double	Untuk menyimpan harga menu

Dari ketiga variabel tersebut, semuanya memiliki akses modifier private, yang artinya variabel tersebut hanya dapat diakses oleh kelas itu sendiri.

2.1.2 Class MenuDatabase

Nama Variabel	Tipe Data	Keterangan
menuItems	ArrayList<MenuItem>	Untuk menyimpan daftar menu

Variabel menuItem bertipe ArrayList bertujuan untuk penyimpanan daftar menu dengan dinamis. ArrayList ini bertipe MenuItem yang merupakan salah satu kelas dari package Menu yang artinya variabel ini bisa mengakses semua elemen dari kelas MenuItem.

2.2 Package Order

Didalam package Order terdapat dua buah kelas yaitu kelas Order dan OrderManager. Berikut ini adalah deskripsi singkat mengenai variabel-variabel yang dibuat di kedua kelas tersebut.

2.2.1 Class Order

Nama Variabel	Tipe Data	Keterangan
orderId	String	Untuk menyimpan id dari setiap orderan
items	ArrayList<MenuItem>	Untuk menyimpan setiap orderan

Variabel items bertipe ArrayList bertujuan untuk penyimpanan daftar orderan dengan dinamis. ArrayList ini bertipe MenuItem yang merupakan salah satu kelas dari package Menu yang artinya variabel ini bisa mengakses semua elemen dari kelas MenuItem.

2.2.2 Class OrderManager

Nama Variabel	Tipe Data	Keterangan
orders	ArrayList<Order>	Untuk menyimpan orderan

Variabel orders bertipe ArrayList bertujuan untuk penyimpanan daftar orderan dengan dinamis. ArrayList ini bertipe Order yang merupakan salah satu kelas dari package Order yang artinya variabel ini bisa mengakses semua elemen dari kelas Order.

2.3 Package Util

Didalam package Util terdapat satu buah kelas yaitu kelas IdGenerator. Berikut ini adalah deskripsi singkat mengenai variabel-variabel yang dibuat di kelas tersebut.

Nama Variabel	Tipe Data	Keterangan
counter	int	Sebagai variabel increment untuk id

Variabel counter ini bersifat statis yang artinya dia sharing dan dapat diakses langsung dari kelas tanpa perlu membuat objek.

3. Constructor dan Method

Dalam kelas Java, constructor dan method bekerja bersama-sama untuk membuat objek berfungsi sesuai dengan kebutuhan aplikasi. Constructor digunakan untuk inisialisasi awal objek, sementara method digunakan untuk mendefinisikan perilaku objek dan melakukan operasi pada data.

Berikut ini akan dijelaskan setiap constructor dari masing-masing kelas:

3.1 Class MenuItem

Nama Method	Jenis	Keterangan
MenuItem(String id, String name, double price)	Constructor	Untuk menginisialisasi objek dari menuItem
getId()	Function	Sebagai getter untuk atribut ID
getName	Function	Sebagai getter untuk atribut name
getPrice	Function	Sebagai getter untuk attribute price

3.2 Class MenuDatabase

Nama Method	Jenis	Keterangan
MenuDatabase()	Constructor	Untuk menginisialisasi objek dari menudatabase
addMenuItem (MenuItem item)	Prosedur	Untuk menambahkan item menu pada database
removeMenuItem (String id)	Function	Fungsi untuk menghapus item dari database
getMenuItems	Function	Sebagai getter untuk attribute menuItems
findMenuItemById (String id)	Function	Untuk mencari menu berdasarkan id
findMenuItemByName (String name)	Function	Untuk mencari menu berdasarkan nama

3.3 Class Order

Nama Method	Jenis	Keterangan
Order()	Constructor	Untuk menginisialisasi objek dari kelas Order
addItem (MenuItem item)	Prosedur	Untuk menambahkan item menu pada orderan
getOrderId	Function	Sebagai getter untuk atribut orderId
getItems	Function	Sebagai getter untuk attribute items

3.4 Class OrderManager

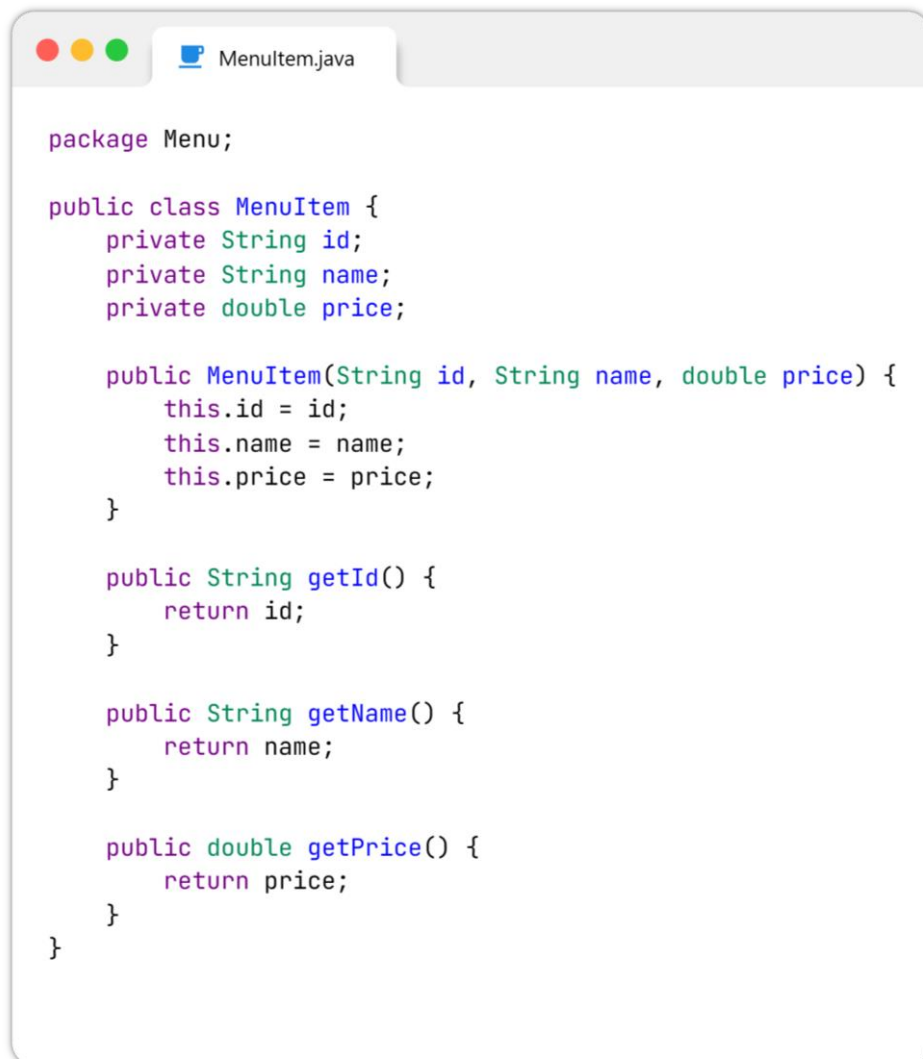
Nama Method	Jenis	Keterangan
OrderManager()	Constructor	Untuk menginisialisasi objek dari orderManager
createOrder (Order order)	Prosedur	Untuk membuat orderan
removeOrder (String orderId)	Function	Untuk menghapus orderan
getOrders()	Function	Sebagai getter untuk attribute orders
getOrderById (String orderId)	Function	Untuk mencari order berdasarkan id

3.5 Class IdGenerator

Nama Method	Jenis	Keterangan
generatorUniqueId	Function	Untuk mengenerate unique id

4. Dokumentasi dan Pembahasan Code

4.1 Class MenuItem



```
package Menu;

public class MenuItem {
    private String id;
    private String name;
    private double price;

    public MenuItem(String id, String name, double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }
}
```

Kode di atas mendefinisikan sebuah kelas dengan nama `MenuItem` di dalam package `Menu`. Berikut adalah penjelasan lebih detailnya:

- `package Menu;`: Mendefinisikan bahwa kelas `MenuItem` berada dalam package `Menu`. Package digunakan untuk mengorganisir kelas-kelas dalam struktur direktori yang terorganisir.
- `public class MenuItem { }`: Mendeklarasikan kelas Java dengan nama `MenuItem`. Kelas ini memiliki tiga atribut: `id`, `name`, dan `price`.
- `private String id; private String name; private double price;`: Mendeklarasikan tiga variabel instance `id`, `name`, dan `price` dengan akses modifier `private`, yang berarti hanya dapat diakses oleh kelas `MenuItem` itu sendiri dan tidak oleh kelas lain.

- `public MenuItem(String id, String name, double price) { }`: Mendefinisikan constructor (konstruktor) untuk kelas `MenuItem` dengan tiga parameter: `id` (bertipe `String`), `name` (bertipe `String`), dan `price` (bertipe `double`). Constructor digunakan untuk menginisialisasi objek baru dari kelas `MenuItem`.
- `this.id = id; this.name = name; this.price = price;`: Menginisialisasi nilai variabel instance `id`, `name`, dan `price` dengan nilai yang diterima melalui parameter constructor.
- `public String getId() { return id; }`: Method `getId()` digunakan untuk mengembalikan nilai dari variabel `id`. Karena variabel `id` bersifat `private`, maka kita menggunakan method ini untuk mengaksesnya dari luar kelas `MenuItem`.
- `public String getName() { return name; }`: Method `getName()` digunakan untuk mengembalikan nilai dari variabel `name`. Sama seperti sebelumnya, method ini digunakan karena variabel `name` bersifat `private`.
- `public double getPrice() { return price; }`: Method `getPrice()` digunakan untuk mengembalikan nilai dari variabel `price`. Method ini juga digunakan untuk mengakses variabel `private price` dari luar kelas `MenuItem`.

4.2 Class MenuDatabase

```
package Menu;

import java.util.ArrayList;

public class MenuDatabase {
    private ArrayList<MenuItem> menuItems;

    public MenuDatabase() {
        menuItems = new ArrayList<>();
    }

    public void addMenuItem(MenuItem item) {
        if (findMenuItemById(item.getId()) != null) {
            System.out.println("Item dengan ID " + item.getId() + " sudah ada pada Menu. Gagal menambahkan menu " + item.getId());
            return;
        }
        menuItems.add(item);
    }

    public boolean removeMenuItem(String id) {
        for (int i = 0; i < menuItems.size(); i++) {
            MenuItem item = menuItems.get(i);
            if (item.getId() == id) {
                menuItems.remove(i);
                return true;
            }
        }
        return false;
    }

    public ArrayList<MenuItem> getMenuItems() {
        return new ArrayList<>(menuItems);
    }

    public MenuItem findMenuItemById(String id){
        for (MenuItem item : menuItems){
            if (item.getId() == id){
                return item;
            }
        }
        return null;
    }

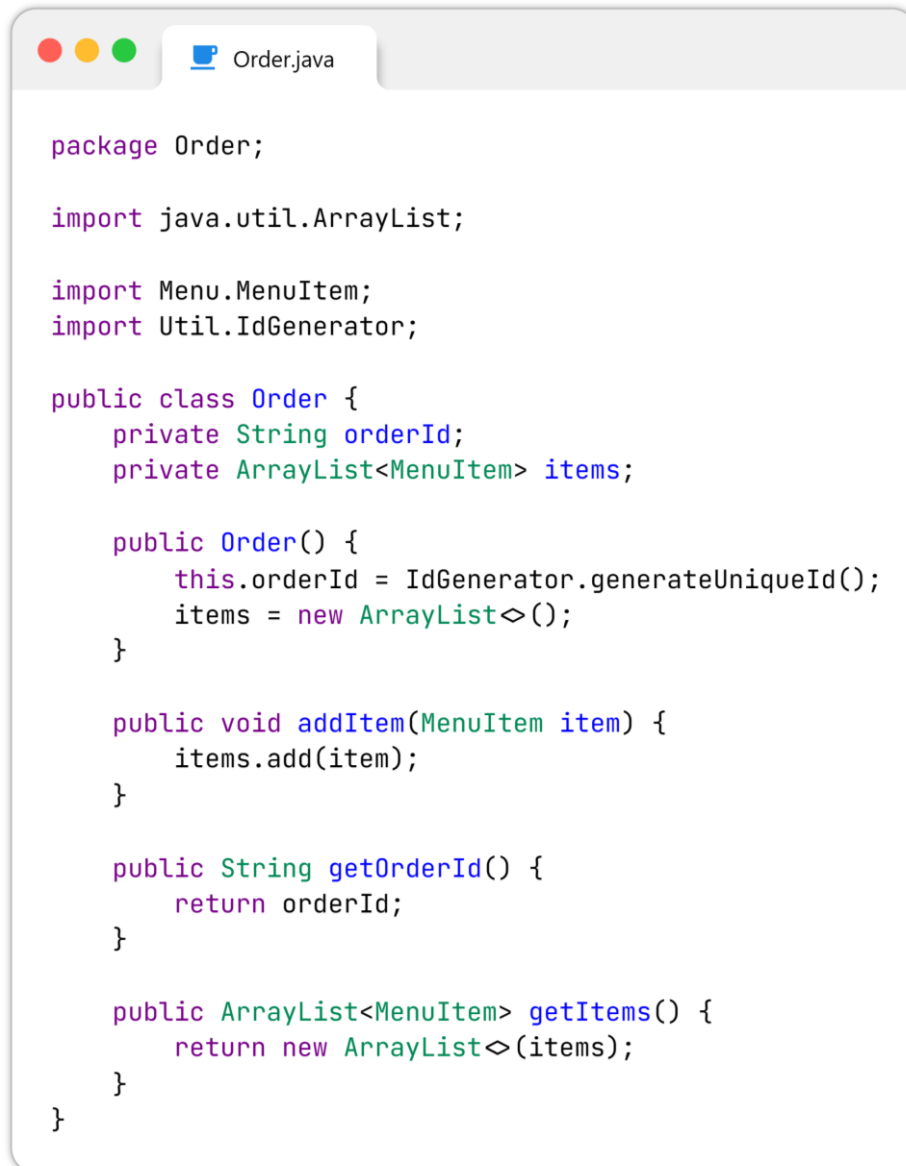
    public MenuItem findMenuItemByName(String name){
        for (MenuItem item : menuItems) {
            if (item.getName() == name) {
                return item;
            }
        }
        return null;
    }
}
```

Ini adalah kode untuk kelas `MenuDatabase` di dalam package `Menu`. Berikut adalah penjelasan lebih detailnya:

- `package Menu;`: Mendefinisikan bahwa kelas `MenuDatabase` berada dalam package `Menu`.
- `import java.util.ArrayList;`: Mengimpor kelas `ArrayList` dari paket `java.util`, yang digunakan untuk menyimpan daftar objek `MenuItem`.
- `private ArrayList<MenuItem> menuItems;`: Mendeklarasikan sebuah variabel instance bernama `menuItems` yang merupakan sebuah `ArrayList` yang berisi objek `MenuItem`. Variabel ini akan digunakan untuk menyimpan daftar item-menu.

- `public MenuDatabase() { }`: Mendefinisikan constructor (konstruktor) untuk kelas `MenuDatabase`. Constructor ini akan membuat sebuah objek `ArrayList` baru saat objek `MenuDatabase` dibuat.
- `public void addItem(MenuItem item) { }`: Method `addItem` digunakan untuk menambahkan sebuah objek `MenuItem` ke dalam daftar `menuItems`. Method ini juga melakukan pengecekan apakah item-menu dengan id yang sama sudah ada sebelumnya dalam daftar. Jika sudah, maka akan ditampilkan pesan kesalahan.
- `public boolean removeMenuItem(String id) { }`: Method `removeMenuItem` digunakan untuk menghapus sebuah objek `MenuItem` dari daftar `menuItems` berdasarkan id yang diberikan.
- `public ArrayList<MenuItem> getMenuItems() { }`: Method `getMenuItems` digunakan untuk mengembalikan salinan daftar `menuItems`. Karena `menuItems` bersifat private, method ini digunakan untuk mengaksesnya dari luar kelas `MenuDatabase`.
- `public MenuItem findMenuItemById(String id) { }`: Method `findMenuItemById` digunakan untuk mencari sebuah objek `MenuItem` berdasarkan id yang diberikan. Jika ditemukan, method ini akan mengembalikan objek `MenuItem` tersebut. Jika tidak ditemukan, maka akan mengembalikan `null`.
- `public MenuItem findMenuItemByName(String name) { }`: Method `findMenuItemByName` digunakan untuk mencari sebuah objek `MenuItem` berdasarkan nama yang diberikan. Jika ditemukan, method ini akan mengembalikan objek `MenuItem` tersebut. Jika tidak ditemukan, maka akan mengembalikan `null`.

4.3 Class Order

A screenshot of a code editor window titled 'Order.java'. The code is written in Java and defines a class 'Order' within the 'Order' package. It imports 'java.util.ArrayList', 'Menu.MenuItem', and 'Util.IdGenerator'. The class has two private attributes: 'orderId' (String) and 'items' (ArrayList<MenuItem>). It includes a constructor that initializes 'orderId' using 'IdGenerator.generateUniqueId()' and 'items' as a new 'ArrayList'. There are three public methods: 'addItem(MenuItem item)' which adds an item to the 'items' list, 'getOrderId()' which returns the 'orderId', and 'getItems()' which returns a new 'ArrayList' containing the 'items' list.

```
package Order;

import java.util.ArrayList;

import Menu.MenuItem;
import Util.IdGenerator;

public class Order {
    private String orderId;
    private ArrayList<MenuItem> items;

    public Order() {
        this.orderId = IdGenerator.generateUniqueId();
        items = new ArrayList<>();
    }

    public void addItem(MenuItem item) {
        items.add(item);
    }

    public String getOrderId() {
        return orderId;
    }

    public ArrayList<MenuItem> getItems() {
        return new ArrayList<>(items);
    }
}
```

Ini adalah kelas `Order` di dalam package `Order`. Berikut adalah penjelasan lebih detailnya:

- `package Order;`: Mendefinisikan bahwa kelas `Order` berada dalam package `Order`.
- `import java.util.ArrayList;`: Mengimpor kelas `ArrayList` dari paket `java.util`, yang digunakan untuk menyimpan daftar objek `MenuItem`.
- `import Menu.MenuItem;`: Mengimpor kelas `MenuItem` dari package `Menu`. Kelas `MenuItem` akan digunakan sebagai bagian dari daftar item-menu dalam sebuah pesanan.

- `import Util.IdGenerator;`: Mengimpor kelas `IdGenerator` dari package `Util`. Kelas `IdGenerator` akan digunakan untuk menghasilkan ID unik untuk setiap pesanan.
- `private String orderId;`: Mendeklarasikan sebuah variabel instance bertipe `String` dengan nama `orderId`, yang akan digunakan untuk menyimpan ID unik dari pesanan.
- `private ArrayList<MenuItem> items;`: Mendeklarasikan sebuah variabel instance bertipe `ArrayList` yang berisi objek `MenuItem`. Variabel ini akan digunakan untuk menyimpan daftar item-menu yang ada dalam pesanan.
- `public Order() { }`: Mendefinisikan constructor (konstruktor) untuk kelas `Order`. Constructor ini akan membuat sebuah objek `ArrayList` baru saat objek `Order` dibuat, serta menginisialisasi `orderId` dengan menggunakan metode `generateUniqueId()` dari kelas `IdGenerator`.
- `public void addItem(MenuItem item) { }`: Method `addItem` digunakan untuk menambahkan sebuah objek `MenuItem` ke dalam daftar `items` dalam pesanan.
- `public String getOrderId() { }`: Method `getOrderId` digunakan untuk mengembalikan ID pesanan (`orderId`).
- `public ArrayList<MenuItem> getItems() { }`: Method `getItems` digunakan untuk mengembalikan salinan daftar `items` dalam pesanan. Karena `items` bersifat `private`, method ini digunakan untuk mengaksesnya dari luar kelas `Order`.

4.4 Class OrderManager

```
package Order;

import java.util.ArrayList;

public class OrderManager {
    private ArrayList<Order> orders;

    public OrderManager() {
        orders = new ArrayList<>();
    }

    public void createOrder(Order order) {
        if (findOrderById(order.getOrderId()) != null) {
            System.out.println("Order dengan ID " +
                order.getOrderId() + " sudah ada. Gagal menambahkan order " +
                order.getOrderId());
            return;
        }
        orders.add(order);
    }

    boolean removeOrder(String orderId) {
        for (int i = 0; i < orders.size(); i++) {
            Order order = orders.get(i);
            if (order.getOrderId() == orderId) {
                orders.remove(i);
                return true;
            }
        }
        return false;
    }

    public ArrayList<Order> getOrders() {
        return new ArrayList<>(orders);
    }

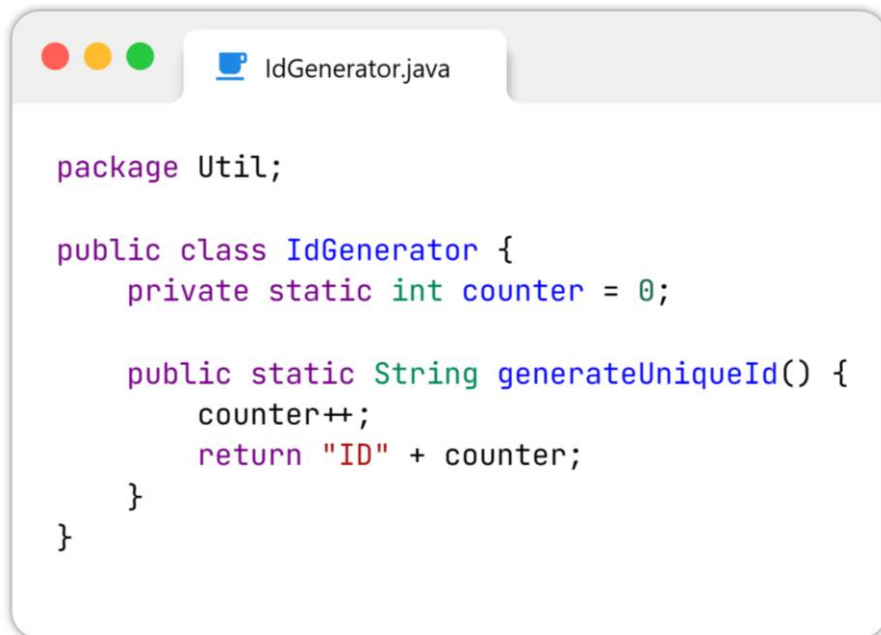
    public Order findOrderById(String orderId){
        for (Order order : orders) {
            if (order.getOrderId() == orderId) {
                return order;
            }
        }
        return null;
    }
}
```

Kode di atas mendefinisikan sebuah kelas dengan nama `OrderManager` di dalam package `Order`. Berikut adalah penjelasan lebih detailnya:

- `package Order;`: Mendefinisikan bahwa kelas `OrderManager` berada dalam package `Order`.

- ``import java.util.ArrayList;``: Mengimpor kelas ``ArrayList`` dari paket ``java.util``, yang digunakan untuk menyimpan daftar objek ``Order``.
- ``private ArrayList<Order> orders;``: Mendeklarasikan sebuah variabel instance bertipe ``ArrayList`` yang berisi objek ``Order``. Variabel ini akan digunakan untuk menyimpan daftar pesanan.
- ``public OrderManager() { }``: Mendefinisikan constructor (konstruktor) untuk kelas ``OrderManager``. Constructor ini akan membuat sebuah objek ``ArrayList`` baru saat objek ``OrderManager`` dibuat.
- ``public void createOrder(Order order) { }``: Method ``createOrder`` digunakan untuk menambahkan sebuah objek ``Order`` ke dalam daftar ``orders`` dalam manajer pesanan. Method ini juga melakukan validasi untuk memastikan bahwa tidak ada pesanan dengan ID yang sama yang sudah ada sebelumnya.
- ``boolean removeOrder(String orderId) { }``: Method ``removeOrder`` digunakan untuk menghapus pesanan berdasarkan ID pesanan yang diberikan. Method ini melakukan iterasi melalui daftar pesanan dan menghapus pesanan yang memiliki ID yang cocok dengan ID yang diberikan.
- ``public ArrayList<Order> getOrders() { }``: Method ``getOrders`` digunakan untuk mengembalikan salinan daftar ``orders`` dalam manajer pesanan. Karena ``orders`` bersifat ``private``, method ini digunakan untuk mengaksesnya dari luar kelas ``OrderManager``.
- ``public Order findOrderById(String orderId) { }``: Method ``findOrderById`` digunakan untuk mencari pesanan berdasarkan ID pesanan yang diberikan. Method ini melakukan iterasi melalui daftar pesanan dan mengembalikan pesanan yang memiliki ID yang cocok dengan ID yang diberikan, atau ``null`` jika tidak ditemukan.

4.5 Class IdGenerator



```
package Util;

public class IdGenerator {
    private static int counter = 0;

    public static String generateUniqueId() {
        counter++;
        return "ID" + counter;
    }
}
```

Kode di atas mendefinisikan kelas `IdGenerator` di dalam package `Util`. Berikut adalah penjelasan lebih detailnya:

- `package Util;`: Mendefinisikan bahwa kelas `IdGenerator` berada dalam package `Util`.
- `public class IdGenerator { }`: Mendefinisikan sebuah kelas Java dengan nama `IdGenerator`.
- `private static int counter = 0;`: Mendeklarasikan sebuah variabel statis bertipe `int` dengan nama `counter` dan memberikan nilai awal `0`. Variabel ini digunakan untuk menyimpan nilai counter yang digunakan untuk menghasilkan ID unik.
- `public static String generateUniqueId() { }`: Mendefinisikan sebuah method `generateUniqueId` yang mengembalikan sebuah string yang merupakan ID unik. Setiap kali method ini dipanggil, variabel `counter` akan ditambahkan satu, dan kemudian ID yang dihasilkan akan berupa string `"ID" + counter`.

Dengan demikian, kelas `IdGenerator` menyediakan sebuah method `generateUniqueId` yang digunakan untuk menghasilkan ID unik dengan format "ID" diikuti oleh nomor yang terus bertambah setiap kali method dipanggil.

4.6 Main Class

```
import Menu.MenuItem;
import Menu.MenuDatabase;
import Order.Order;
import Order.OrderManager;
// import Util.IdGenerator;

import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        // Membuat objek MenuDatabase
        MenuDatabase menuDatabase = new MenuDatabase();

        // Menambahkan item-menu ke MenuDatabase
        MenuItem menuItem1 = new MenuItem("M1", "Nasi Goreng", 15000);
        MenuItem menuItem2 = new MenuItem("M2", "Mie Ayam", 12000);
        MenuItem menuItem3 = new MenuItem("M3", "Ayam Bakar", 20000);
        MenuItem menuItem4 = new MenuItem("M3", "Nasi Uduk", 20000);

        menuDatabase.addMenuItem(menuItem1);
        menuDatabase.addMenuItem(menuItem2);
        menuDatabase.addMenuItem(menuItem3);
        menuDatabase.addMenuItem(menuItem4);

        // Menampilkan menu
        System.out.println("Menu:");
        ArrayList<MenuItem> menuItems = menuDatabase.getMenuItems();
        for (MenuItem item : menuItems) {
            System.out.println(item.getId() + " - " + item.getName() + " : " + item.getPrice());
        }

        // Membuat objek OrderManager
        OrderManager orderManager = new OrderManager();

        // Menambahkan pesanan
        Order order1 = new Order();
        order1.addItem(menuItem1);
        Order order2 = new Order();
        order2.addItem(menuItem2);

        orderManager.createOrder(order1);
        orderManager.createOrder(order2);
        orderManager.createOrder(order2);

        // Menampilkan pesanan
        System.out.println("\nDaftar Pesanan:");
        ArrayList<Order> orders = orderManager.getOrders();
        for (Order order : orders) {
            System.out.println("Order ID: " + order.getOrderID());
            ArrayList<MenuItem> orderItems = order.getItems();
            System.out.println("Items:");
            for (MenuItem item : orderItems) {
                System.out.println("- " + item.getName() + " : " + item.getPrice());
            }
        }

        System.out.println();

        // Mencari item menu berdasarkan ID
        String cariIdItem = "M2";
        MenuItem idStatus = menuDatabase.findMenuItemById(cariIdItem);

        if (idStatus != null){
            System.out.println("Id " + cariIdItem + " ditemukan : " + idStatus.getName());
        } else{
            System.out.println("Item dengan Id " + cariIdItem + " tidak ditemukan.");
        }

        System.out.println();

        // Mencari item menu berdasarkan Nama
        String cariNamaItem = "Ayam Bakar";
        MenuItem namaStatus = menuDatabase.findMenuItemByName(cariNamaItem);

        if (namaStatus != null){
            System.out.println("Nama " + cariNamaItem + " ditemukan : " + namaStatus.getName());
        } else{
            System.out.println("Item dengan nama " + cariNamaItem + " tidak ditemukan.");
        }
    }
}
```

Kode di atas merupakan implementasi dari program utama (main program) yang menggunakan kelas-kelas yang telah dibuat sebelumnya, yaitu MenuItem, MenuDatabase, Order, dan OrderManager. Berikut penjelasan lebih detailnya:

- Import package dan kelas yang diperlukan:

```
import Menu.MenuItem;  
import Menu.MenuDatabase;  
import Order.Order;  
import Order.OrderManager;
```

- Mendefinisikan kelas Main sebagai kelas utama:

```
public class Main {
```

- Mendefinisikan method main sebagai method utama yang akan dieksekusi pertama kali saat program dijalankan:

```
public static void main(String[] args) {
```

- Membuat objek MenuDatabase dan menambahkan beberapa item-menu ke dalamnya:

```
MenuDatabase menuDatabase = new MenuDatabase();  
MenuItem menuItem1 = new MenuItem("M1", "Nasi Goreng", 15000);  
MenuItem menuItem2 = new MenuItem("M2", "Mie Ayam", 12000);  
MenuItem menuItem3 = new MenuItem("M3", "Ayam Bakar", 20000);  
MenuItem menuItem4 = new MenuItem("M3", "Nasi Uduk", 20000);  
menuDatabase.addMenuItem(menuItem1);  
menuDatabase.addMenuItem(menuItem2);  
menuDatabase.addMenuItem(menuItem3);  
menuDatabase.addMenuItem(menuItem4);
```

- Menampilkan menu yang telah ditambahkan:

```
System.out.println("Menu:");
ArrayList<MenuItem> menuItems = menuDatabase.getMenuItems();
for (MenuItem item : menuItems) {
    System.out.println(item.getId() + " - " + item.getName() + " : " + item.getPrice());
}
```

- Membuat objek OrderManager dan menambahkan beberapa pesanan:

```
OrderManager orderManager = new OrderManager();
Order order1 = new Order();
order1.addItem(menuItem1);
Order order2 = new Order();
order2.addItem(menuItem2);
orderManager.createOrder(order1);
orderManager.createOrder(order2);
orderManager.createOrder(order2);
```

- Menampilkan daftar pesanan yang telah dibuat:

```
System.out.println("\nDaftar Pesanan:");
ArrayList<Order> orders = orderManager.getOrders();
for (Order order : orders) {
    System.out.println("Order ID: " + order.getOrderID());
    ArrayList<MenuItem> orderItems = order.getItems();
    System.out.println("Items:");
    for (MenuItem item : orderItems) {
        System.out.println("- " + item.getName() + " : " + item.getPrice());
    }
}
```

- Mencari item menu berdasarkan ID dan menampilkan hasilnya:

```
String cariIdItem = "M2";
MenuItem idStatus = menuDatabase.findMenuItemById(cariIdItem);
if (idStatus != null){
    System.out.println("Id " + cariIdItem + " ditemukan : " + idStatus.getName());
} else{
    System.out.println("Item dengan Id " + cariIdItem + " tidak ditemukan.");
}
```

- Mencari item menu berdasarkan nama dan menampilkan hasilnya:

```
String cariNamaItem = "Ayam Bakar";
MenuItem namaStatus = menuDatabase.findMenuItemByName(cariNamaItem);
if (namaStatus != null){
    System.out.println("Nama " + cariNamaItem + " ditemukan : " + namaStatus.getName());
} else{
    System.out.println("Item dengan nama " + cariNamaItem + " tidak ditemukan.");
}
```

Dengan demikian, kode di atas adalah sebuah contoh lengkap dari penggunaan kelas-kelas yang telah dibuat sebelumnya untuk mengelola menu dan pesanan pada sistem pemesanan makanan online.

5. Kesimpulan

Dalam proyek ini, kita telah mengembangkan sebuah sistem pemesanan makanan online yang terdiri dari beberapa kelas dan package yang berbeda. Berikut adalah kesimpulan dari proyek ini:

- **Penggunaan Kelas dan Package:** Kita menggunakan konsep package dalam Java untuk mengorganisir kelas-kelas ke dalam kelompok yang terkait. Setiap package seperti `'Menu'`, `'Order'`, dan `'Util'` memiliki tanggung jawabnya masing-masing dalam sistem.
- **Implementasi Kelas-Kelas:** Kita mengimplementasikan beberapa kelas yang saling berinteraksi dalam sistem, termasuk `'MenuItem'`, `'MenuDatabase'`, `'Order'`, dan `'OrderManager'`. Setiap kelas memiliki peranannya dalam sistem, seperti menyimpan detail menu, mengelola pesanan, dan menghasilkan ID unik.
- **Penggunaan ArrayList:** Untuk menyimpan daftar item menu dan pesanan, kita menggunakan struktur data `ArrayList`. Hal ini memudahkan dalam penambahan, penghapusan, dan pencarian elemen-elemen dalam daftar.
- **Validasi dan Penanganan Error:** Kita telah menambahkan validasi dalam beberapa method untuk menangani situasi seperti duplikasi ID saat menambahkan item baru ke menu atau membuat pesanan baru. Selain itu, kita juga memberikan pesan error yang jelas jika operasi tidak berhasil.
- **Penggunaan Constructor dan Method:** Constructor digunakan untuk inisialisasi objek saat objek tersebut dibuat, sedangkan method digunakan untuk melakukan operasi tertentu pada objek. Dalam proyek ini, constructor digunakan untuk menginisialisasi objek `'Order'` dan `'MenuDatabase'`, sedangkan method-method lainnya digunakan untuk berbagai operasi seperti menambahkan item menu, membuat pesanan, dan mencari item berdasarkan ID atau nama.
- **Penggunaan Getter dan Setter:** Kita menggunakan getter untuk mendapatkan nilai atribut dari sebuah objek dan setter untuk mengubah nilai atribut tersebut jika diperlukan.
- **Penggunaan Static Method:** Static method digunakan dalam kelas `'IdGenerator'` untuk menghasilkan ID unik secara global tanpa perlu membuat objek kelas `'IdGenerator'`.

Dengan demikian, proyek ini memberikan contoh yang baik tentang bagaimana mengorganisir, mengimplementasikan, dan menggunakan kelas-kelas dalam sebuah sistem berbasis Java.