

FINAL EXAM EVEN SEMESTER ACADEMIC YEAR 2023/2024

INFORMATION SYSTEM STUDY PROGRAM

FACULTY OF ENGINEERING AND INFORMATICS

UNIVERSITAS MULTIMEDIA NUSANTARA

OBJECT ORIENTED PROGRAMMING LEC + LAB

Group Member:

- Darrell Jeremy (00000093506)
- Taura Aydinarafa Khairan (00000093612)
- Shining Sunny Chang (00000094693)
- Eileen Angelina (00000095167)

Mata Kuliah: Object Oriented Programming - C

Final Project Assignment, Final Exam Theory and Final Exam Lab scores will be integrated. Pay attention to the scoring rubric to get the best score.

1. Question 1: Sub-CLO 1, Weight (13.76%)

In the Final Project report, detail and narrate the following points:

- a. Background for case study selection (problem and source references)
- b. Design solutions using an OOP approach and linked list data structure
- c. The Use of OOP concepts in selected study cases (Inheritance, Polymorphism, Abstraction, Encapsulation)

Answer:

Proyek akhir kami adalah tentang restoran yang bernama Five Spices. Nama dari restoran ini memiliki arti bahwa kami menggunakan campuran bumbu rempah yang beragam. Kami memilih tema restoran untuk proyek ini karena ingin mengoptimalkan efisiensi sistem dalam berbagai aspek manajemen restoran, seperti manajemen stok, manajemen data, dan informasi penjualan, sehingga operasional berjalan lancar dan pengelolaan transaksi menjadi lebih baik, akurat, dan efisien. Dengan menggunakan Object-Oriented Programming (OOP), kami dapat menciptakan sistem manajemen restoran yang lebih reusable dan dapat digunakan kembali

dalam kelas-kelas lain, seperti manajemen pesanan dan stok, sehingga mengurangi pengulangan kode. Pendekatan ini memungkinkan fokus pada bagian-bagian tertentu dari sistem tanpa perlu memodifikasi seluruh kode, yang pada gilirannya meningkatkan efisiensi pengembangan dan pemeliharaan sistem.

OOP memungkinkan representasi dunia nyata yang lebih akurat dari objek-objek yang ada dalam restoran, seperti Pelanggan, Menu Item, Pesanan, Karyawan, Stok Makanan, Koki, dan lain-lain. Dalam OOP, kita dapat membuat kelas yang mewakili objek-objek ini sehingga kode menjadi lebih mudah dipelihara. Selain itu, OOP menyediakan enkapsulasi dan perlindungan data melalui atribut dan metode. Misalnya, objek Pesanan memiliki metode untuk menghitung total tagihan, di mana hanya metode yang diotorisasi yang dapat mengubah detail pesanan. Implementasi OOP juga mendukung pewarisan dan polimorfisme, memungkinkan kita untuk membuat kelas turunan seperti Appetizer, Main Course, Dessert, dan Drinks, serta metode 'Override' yang identik dalam kelas induk atau antarmuka. Misalnya, kita dapat menurunkan subkelas Chef dengan fungsi khusus. Pendekatan ini mengurangi duplikasi kode dan meningkatkan fleksibilitas.

Di dalam sistem restoran Five Spices ini, kami juga menerapkan berbagai konsep OOP, mulai dari Class & Object, Encapsulation, Inheritance, Interface, Abstraction, Polymorphism, hingga Java Collection Framework, Single Linked List, Doubly Linked List, Single Circular Linked List, Circular Double Linked List, Stack, dan Queue. Dengan penerapan konsep-konsep OOP ini, kami memastikan bahwa semua alur operasional restoran bekerja sesuai dengan yang diharapkan, sehingga meningkatkan efisiensi dan efektivitas sistem manajemen restoran secara keseluruhan.

Kerangka Code :



Di dalam Kerangka kode kami, akan ada beberapa kelas yaitu : Restaurant, Menu, MenuCDLLList, MenuStock, Appetizer, MainCourse, Dessert, Drinks, Iprepare, Iservefood, Koki, KokiShift, ShiftSCLList, ActivityLog, Customer, FiveSpicesQueue, Table, dan Pesanan.

Untuk Atribut di dalam Kerangka kode kami adalah :

- **Restaurant** : String location, String name, int year, String owner, double rating, String nomorMeja dan String status, Feedback feedbackStack
 - Feedback: Node top; (Inner Class Restaurant)
 - Node: String feedback, Node next; (Inner Class Feedback)
- **Menu** : String nama, String harga, double rating, String type, String isFavorite;
- **MenuCDLLList** : MenuItem head, MenuItem current;

- MenuItem : MenuItem menu, MenuItem next, MenuItem prev; (Inner Class MenuCDLList)
- **MenuStock** : HashMap<String, Integer> stock, int currentStock;
- **Koki** : String name, String position, String speciality, boolean isCertified, int experience, MenuStock menuStock;
- Private static Koki head, Koki next (Inner Class Koki)
- **KokiShift** : String name, String position, String speciality, int experience, boolean isCertified, MenuStock menuStock, String shiftWaktu;
- **ShiftSCLList** : Node current, Node head, int size;
 - Node : ShiftKoki data, Node next; (Inner Class ShiftSCLList)
- **ActivityLog** : int day, int customers, double revenue, LogNode next, LogNode prev;
 - LogNode : LogNode head, LogNode tail, LogNode current; (Inner Class ActivityLog)
- **Customer** : String nameCust, int nomorAntrian, boolean isVip;
- **FiveSpicesQueue** : Node front, Node rear
 - Node : Customer customer, Node next; (Inner Class FiveSpicesQueue)
- **Table** : String nomorMeja, String status, ArrayList<Table> tables, Queue<String> antrian, int nomorAntrian
- **Pesanan** : String nama, double harga, int jumlah;

Konstruktor :

Konstruktor adalah metode khusus yang digunakan untuk menginisialisasi objek saat dibuat. Dalam kode di atas, terdapat beberapa konstruktor dalam berbagai kelas. Mari kita lihat berbagai macam konstruktor yang ada dan memberikan penjelasan untuk masing-masing:

Kelas	Parameter Konstruktor	Deskripsi Parameter
Restaurant	String name, String location, int year, String owner, double rating	Berisikan Informasi Restoran seperti Nama, lokasi, tahun, pemilik, rating restoran
Feedback	-	-
Node	String feedback	Digunakan untuk menyimpan pesan feedback yang diberikan saat pembuatan objek Node.
Menu	String nama, String harga, double rating, String type, String isFavorite	Digunakan untuk menaruh informasi menu seperti Nama, harga, rating, tipe, dan apakah favorit.
Appetizer	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu Appetizer Nama, harga, rating, tipe, apakah favorit
MainCourse	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu MainCourse Nama, harga, rating, tipe, apakah favorit
Dessert	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu Dessert Nama, harga, rating, tipe, apakah favorit
Drinks	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu Drinks Nama, harga, rating, tipe, apakah favorit
MenuItem	Menu menu	Digunakan untuk menginisialisasi objek MenuItem dengan menu yang diberikan.
MenuStock	-	-
Koki	String name, String position, String speciality, int experience, boolean isCertified, MenuStock menuStock	Berisikan identitas koki itu sendiri meliputi nama, posisi, pengalaman, apakah koki tersebut bersertifikasi atau tidak dan stok menu

KokiShift	String name, String position, String speciality, int experience, boolean isCertified, MenuStock menuStock, String shiftWaktu	Class turunan dari Koki berisikan Nama, posisi, spesialisasi, pengalaman, apakah bersertifikat, stok menu, dan waktu shift koki
Node	ShiftKoki data	Digunakan untuk menyimpan data berupa objek ShiftKoki yang diberikan saat pembuatan objek Node.
ActivityLog	-	-
LogNode	Int day, int customers, double revenue	Terdiri atas atribut hari, jumlah pelanggan dan pendapatan
Customer	String nameCust, int nomorAntrian, boolean isVip	Terdiri atas data pelanggan berisi nama pelanggan, nomor antrian yang dimiliki dan apakah pelanggan termasuk pelanggan Vip atau tidak
Node	Customer customer	Digunakan untuk menyimpan data Customer yang diberikan saat pembuatan objek Node.
Table	String nomorMeja, String status	Terdiri atas nomor meja yang pelanggan dapat begitu juga dengan status meja apakah sudah dipesan atau tidak
Pesanan	String nama, double harga, int jumlah	Terdiri atas nama pesanan dengan harga dan jumlah pesanan

Sumber: [Class vs. Object in OOP | Baeldung on Computer Science](#)

Implementasi Code :

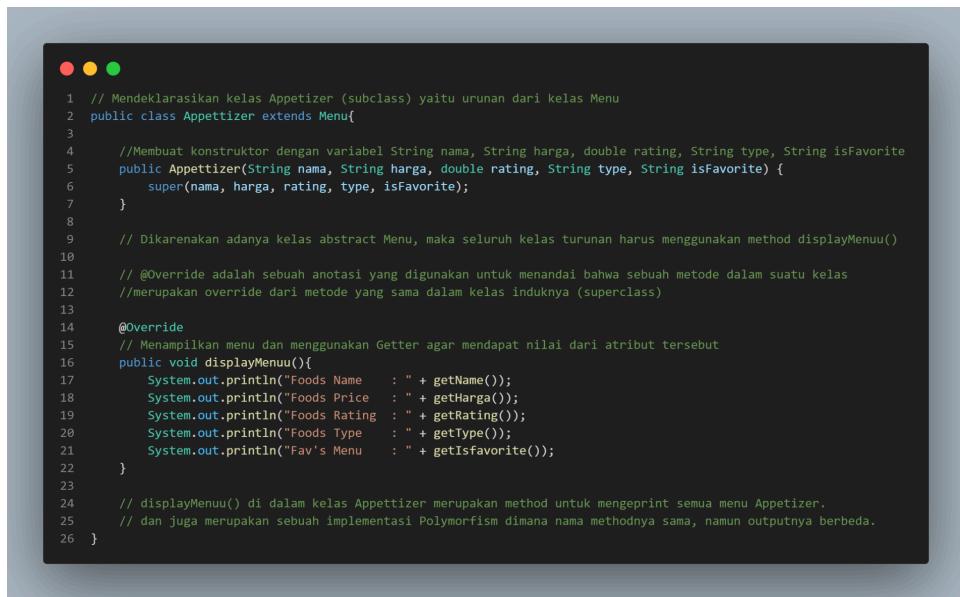
Inheritance (warisan) dalam Java memungkinkan kelas untuk mewarisi metode dan atribut dari kelas lain. Kelas yang memberikan warisan disebut kelas induk atau superclass, dan kelas yang menerima warisan disebut kelas turunan atau subclass.

Inheritance memungkinkan hierarki kelas yang memungkinkan untuk mengatur dan memodelkan hubungan antar kelas dalam program Java. Ini karena kelas turunan dapat menambahkan atribut dan metode tambahan atau mengganti perilaku yang ada sesuai dengan kebutuhan aplikasi. Untuk melakukan pewarisan, kata kunci ‘extends’ digunakan untuk menunjukkan kelas mana yang merupakan kelas induk. Dengan meng-extends kelas, kelas turunan dapat mewarisi karakteristik dan tingkah laku dari kelas induk. Di dalam kode kami,

kami menggunakan kelas Menu sebagai Parent Class dan turunannya yaitu Appetizer, MainCourse, Dessert, dan Drinks. Dikarenakan Appetizer, MainCourse, Dessert, dan Drinks bisa dibilang adalah lanjutan atau turunan dari Menu.

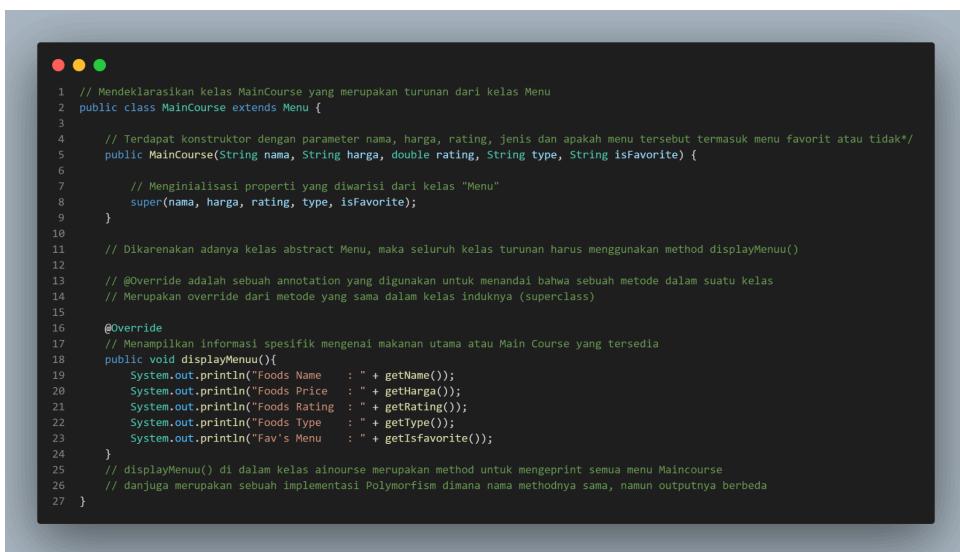
Sumber: <https://www.petanikode.com/java-oop-inheritance/>

Appetizer:



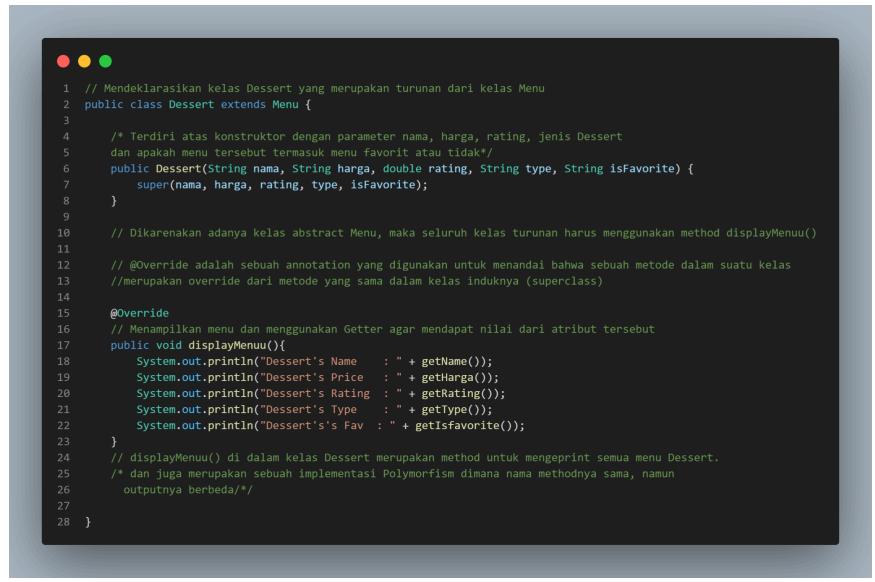
```
1 // Mendeklarasikan kelas Appetizer (subclass) yaitu urunan dari kelas Menu
2 public class Appetizer extends Menu{
3
4     //Membuat konstruktor dengan variabel String nama, String harga, double rating, String type, String isFavorite
5     public Appetizer(String nama, String harga, double rating, String type, String isFavorite) {
6         super(nama, harga, rating, type, isFavorite);
7     }
8
9     // Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
10
11    // @Override adalah sebuah anotasi yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
12    // merupakan override dari metode yang sama dalam kelas induknya (superclass)
13
14    @Override
15    // Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
16    public void displayMenu(){
17        System.out.println("Foods Name : " + getName());
18        System.out.println("Foods Price : " + getharga());
19        System.out.println("Foods Rating : " + getRating());
20        System.out.println("Foods Type : " + getType());
21        System.out.println("Fav's Menu : " + getisfavorite());
22    }
23
24    // displayMenu() di dalam kelas Appetizer merupakan method untuk mengeprint semua menu Appetizer,
25    // dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda.
26 }
```

MainCourse:



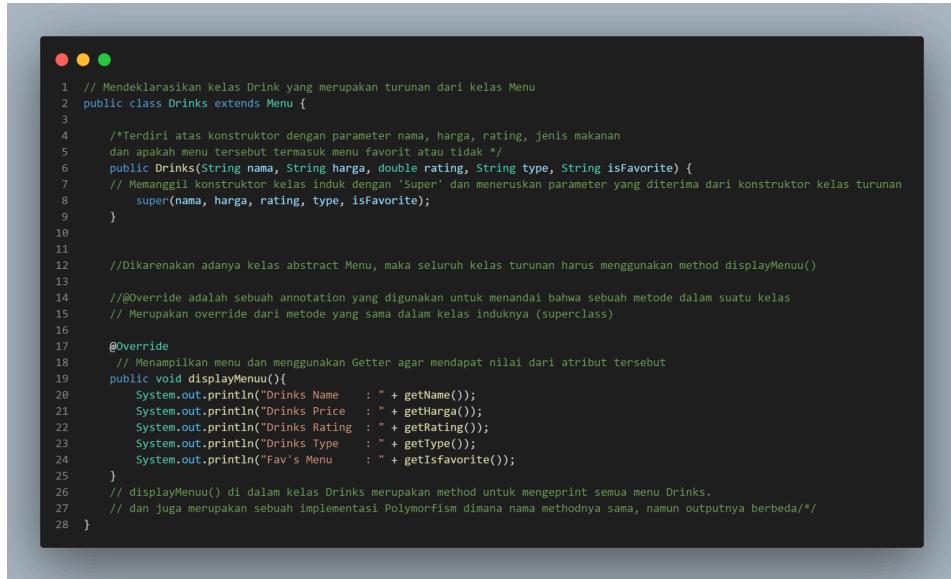
```
1 // Mendeklarasikan kelas MainCourse yang merupakan turunan dari kelas Menu
2 public class MainCourse extends Menu {
3
4     // Terdapat konstruktor dengan parameter nama, harga, rating, jenis dan apakah menu tersebut termasuk menu favorit atau tidak*
5     public MainCourse(String nama, String harga, double rating, String type, String isFavorite) {
6
7         // Menginisialisasi properti yang diwarisi dari kelas "Menu"
8         super(nama, harga, rating, type, isFavorite);
9     }
10
11    // Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
12
13    // @Override adalah sebuah annotation yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
14    // Merupakan override dari metode yang sama dalam kelas induknya (superclass)
15
16    @Override
17    // Menampilkan informasi spesifik mengenai makanan utama atau Main Course yang tersedia
18    public void displayMenu(){
19        System.out.println("Foods Name : " + getName());
20        System.out.println("Foods Price : " + getHarga());
21        System.out.println("Foods Rating : " + getRating());
22        System.out.println("Foods Type : " + getType());
23        System.out.println("Fav's Menu : " + getIsfavorite());
24    }
25    // displayMenu() di dalam kelas Maincourse merupakan method untuk mengeprint semua menu Maincourse
26    // dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda
27 }
```

Dessert :



```
1 // Mendeklarasikan kelas Dessert yang merupakan turunan dari kelas Menu
2 public class Dessert extends Menu {
3
4     /* Terdiri atas konstruktor dengan parameter nama, harga, rating, jenis Dessert
5      dan apakah menu tersebut termasuk menu favorit atau tidak*/
6     public Dessert(String nama, String harga, double rating, String type, String isFavorite) {
7         super(nama, harga, rating, type, isFavorite);
8     }
9
10    // Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
11
12    // @Override adalah sebuah annotation yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
13    // merupakan override dari metode yang sama dalam kelas induknya (superclass)
14
15    @Override
16    // Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
17    public void displayMenu(){
18        System.out.println("Dessert's Name : " + getName());
19        System.out.println("Dessert's Price : " + getHarga());
20        System.out.println("Dessert's Rating : " + getRating());
21        System.out.println("Dessert's Type : " + getType());
22        System.out.println("Dessert's Fav : " + getIsfavorite());
23    }
24    // displayMenu() di dalam kelas Dessert merupakan method untuk mengeprint semua menu Dessert.
25    /* dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun
26     outputnya berbeda*/
27
28 }
```

Drinks:



```
1 // Mendeklarasikan kelas Drink yang merupakan turunan dari kelas Menu
2 public class Drinks extends Menu {
3
4     /*Terdiri atas konstruktor dengan parameter nama, harga, rating, jenis makanan
5      dan apakah menu tersebut termasuk menu favorit atau tidak */
6     public Drinks(String nama, String harga, double rating, String type, String isFavorite) {
7         // Memanggil konstruktor kelas induk dengan 'Super' dan meneruskan parameter yang diterima dari konstruktor kelas turunan
8         super(nama, harga, rating, type, isFavorite);
9     }
10
11
12    //Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
13
14    // @Override adalah sebuah annotation yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
15    // Merupakan override dari metode yang sama dalam kelas induknya (superclass)
16
17    @Override
18    // Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
19    public void displayMenu(){
20        System.out.println("Drinks Name : " + getName());
21        System.out.println("Drinks Price : " + getHarga());
22        System.out.println("Drinks Rating : " + getRating());
23        System.out.println("Drinks Type : " + getType());
24        System.out.println("Fav's Menu : " + getIsfavorite());
25    }
26    // displayMenu() di dalam kelas Drinks merupakan method untuk mengeprint semua menu Drinks.
27    // dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda*/
28 }
```

Polymorphism dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “bentuk” method yang berbeda-beda meskipun namanya sama. Dapat dibilang bahwa polymorphism mengacu pada kemampuan suatu objek untuk memiliki berbagai bentuk atau perilaku. Polymorphism memungkinkan penggunaan objek dari kelas tertentu sebagai objek dari kelas yang lebih umum, yang kemudian dapat berperilaku sesuai dengan jenis spesifiknya. Ini meningkatkan fleksibilitas penggunaan objek dan memungkinkan untuk menangani berbagai situasi dengan cara yang lebih umum. Di dalam code kami, kami

menggunakan “@Override” untuk method DisplayMenuu() untuk setiap subclass, yang dimana tiap method akan menyajikan output yang berbeda.

Sumber: [Belajar Java OOP: Memahami Prinsip Polimorfisme dalam OOP \(petanikode.com\)](https://petanikode.com)

```
// Terdapat method displayMenuu() yang akan digunakan kepada subclass Menu nanti
public void displayMenuu() {

}
```

```
@Override
// Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
public void displayMenuu(){
    System.out.println("Foods Name      : " + getName());
    System.out.println("Foods Price     : " + getHarga());
    System.out.println("Foods Rating   : " + getRating());
    System.out.println("Foods Type     : " + getType());
    System.out.println("Fav's Menu      : " + getIsfavorite());
}
```

```
@Override
// Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
public void displayMenuu(){
    System.out.println("Drinks Name     : " + getName());
    System.out.println("Drinks Price    : " + getHarga());
    System.out.println("Drinks Rating  : " + getRating());
    System.out.println("Drinks Type    : " + getType());
    System.out.println("Fav's Menu      : " + getIsfavorite());
}
// displayMenuu() di dalam kelas Drinks merupakan method untuk mengeprint semua menu Drinks.
// dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda/*
```

Abstraction adalah suatu prinsip dalam pemrograman OOP yang digunakan untuk menghilangkan atau menghapus karakteristik dari suatu objek agar mengurangi kompleksitasnya. Sederhananya, abstraksi memungkinkan kita untuk menemukan dan mendefinisikan perilaku objek tanpa memperhatikan implementasi khusus mereka. Dengan menggunakan abstraksi, kita dapat membuat kelas yang mewakili ide-ide tingkat tinggi yang kemudian dapat diimplementasikan dalam kelas yang lebih khusus. Ini memungkinkan untuk membuat model yang lebih sederhana dan mudah dipahami untuk mengatasi kompleksitas pengembangan perangkat lunak. Dalam Java, kelas abstrak dan interface biasanya digunakan untuk menerapkan abstraksi. Kelas abstrak adalah kelas yang tidak dapat diinstansiasi dan memiliki metode abstrak, yang harus diterapkan oleh kelas-kelas turunannya. Di dalam kode kami, class Menu akan menjadi class abstract. Class Menu akan memaksa kelas turunannya yaitu Appetizer, MainCourse, Dessert, dan Drinks untuk melakukan method displayMenuu().

Sumber: [Abstraction adalah: Pengertian dan Analogi Sederhananya | kumparan.com](https://www.kumparan.com)



```
1 // Mendeklarasikan kelas Abstract Menu
2 public abstract class Menu {
3     // Terdiri atas atribut privat String nama, String harga, double rating, String type, String isFavorite
4     private String nama;
5     private String harga;
6     private double rating;
7     private String type;
8     private String isFavorite;
9
10 /*String nama digunakan untuk nama menu, String digunakan untuk harga menu,
11 double rating digunakan untuk rating sebuah menu dari 1-5, String type digunakan untuk mengetahui jenis menu,
12 String isFavorite digunakan untuk mengetahui apakah menu itu favorit atau bukan*/
13
14 // Terdapat konstruktor dengan variabel String nama, String harga, double rating, String type, String isFavorite
15     public Menu (String nama , String harga, double rating, String type, String isFavorite){
16         this.nama = nama ;
17         this.harga = harga;
18         this.rating = rating;
19         this.type = type;
20         this.isFavorite = isFavorite;
21     }
22
23 // Menggunakan method Getter
24     public String getName () {
25         return nama;
26     }
27
28     public String getHarga(){
29         return harga;
30     }
31
32     public double getRating (){
33         return rating;
34     }
35
36     public String getType(){
37         return type;
38     }
39
40     public String getIsfavorite (){
41         return isFavorite;
42     }
43
44 // Menggunakan method Setter
45     public void setIsfavorite (String isFavorite) {
46         this.isFavorite = isFavorite;
47     }
48
49 // Terdapat method displayMenuu() yang akan digunakan kepada subclass Menu nanti
50     public void displayMenuu() {
51
52     }
53
54 // Terdapat method displayMenuWithBorders() yang akan digunakan di Main.
55 /*displayMenuWithBorders() merupakan method untuk menampilkan menu dengan
56 border agar saat user menampilkan menu, menu yang ditampilkan itu lebih rapi*/
57
58 }
```

Encapsulation adalah salah satu prinsip dasar dalam Object Oriented Programming (OOP) yang bertujuan untuk membungkus data dalam satu unit. Dalam pemrograman berorientasi objek (OOP), encapsulation adalah konsep penting yang bertujuan untuk menyatukan / membungkus data (atribut) dan fungsi (metode) yang terkait ke dalam sebuah unit tunggal

yang disebut class. Dengan adanya encapsulation, detail internal implementasi objek disembunyikan dari program lain dan dunia luar, sehingga yang terlihat sekumpulan metode public yang bisa digunakan untuk berinteraksi dengan objek tersebut. Dengan adanya encapsulation, maka ada yang kita buat akan lebih aman. Dengan membatasi akses langsung ke atribut, kita bisa melindungi data dari perubahan yang tidak diinginkan. Encapsulation juga membuat kode lebih terorganisir dan modular, sehingga lebih mudah dipahami, dipelihara, dan digunakan kembali. Di dalam atribut kode kami, kami membuat semua atribut berstatus private. Dan mengaksesnya menggunakan getter setter method.

Sumber: [Apa itu Encapsulation? Pengertian dan contoh 2023 | RevoU](#)

```
// Mendeklarasikan kelas Abstract Menu
public abstract class Menu {
    // Terdiri atas atribut privat String nama, String harga, double rating, String type, String isFavorite
    private String nama;
    private String harga;
    private double rating;
    private String type;
    private String isFavorite;

    // Menggunakan method Getter
    public String getName () {
        return nama;
    }

    public String getHarga(){
        return harga;
    }

    public double getRating (){
        return rating;
    }

    public String getType(){
        return type;
    }

    public String getIsfavorite (){
        return isFavorite;
    }
}
```

Question 2: Sub-CLO 2, Weight(30%)

In the Final Project report, create a detailed mapping of what and how the OOP concept is used in the selected study case. Explain how many objects are formed, and how the objects that act as "Transactions" work.

Answer:

a. Objects:

- `ArrayList<Pesanan> pesananList = new ArrayList<>();`

Object ini berfungsi untuk menyimpan List list pesanan yang akan dipesan oleh user.

- `Table table = new Table();`

Object ini berfungsi untuk mendeklarasikan meja - meja yang dapat di reserve oleh user

- `MenuCDLList menuListAp = new MenuCDLList();`
- `MenuCDLList menuListM = new MenuCDLList();`
- `MenuCDLList menuListDs = new MenuCDLList();`
- `MenuCDLList menuListDr = new MenuCDLList();`

Object ini berfungsi untuk menyimpan dan menampilkan menu Appetizer, Main Course, Dessert, dan Drink dalam Double Circular Linked List. `menuListAp` menyimpan hanya untuk menu Appetizer. `menuListM` menyimpan hanya untuk menu MainCourse. `menuListDs` menyimpan hanya untuk menu Dessert. `menuListDr` menyimpan hanya untuk menu Drink.

- `ShiftSCLList shiftListKoki = new ShiftSCLList();`

Object ini berfungsi untuk menyimpan dan menampilkan shift dari list koki dalam Single Circular Linked List

- ActivityLog activityLog = new ActivityLog();

Object ini berfungsi untuk menyimpan aktifitas restoran secara random dalam 7 hari kebelakang dan terdapat revenue dan jumlah pelanggan yang datang dan diimplementasikan menggunakan Doubly Circular Linked list

- Appetizer appetizer1 = new Appetizer("Caesar Salad", "Rp.20.000", 4.2, "Appetizer", "Yes");
- Appetizer appetizer2 = new Appetizer("Mozzarella Sticks", "Rp.17.000", 4.3, "Appetizer", "Yes");
- Appetizer appetizer3 = new Appetizer("Bruschetta", "Rp.18.000", 3.2, "Appetizer", "No");
- Appetizer appetizer4 = new Appetizer("Shrimp Cocktail", "Rp.28.000", 3.4, "Appetizer", "No");
- Appetizer appetizer5 = new Appetizer("Meat Risoles", "Rp.5.000", 4.3, "Appetizer", "Yes");

Object ini merupakan nama-nama dan informasi menu khusus Appetizer. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama menu yang ada adalah Caesar Salad, Mozzarella Sticks, Bruschetta, Shrimp Cocktail, dan Meat Risoles dengan harga & rating yang bervariasi.

- MainCourse mainCourse1 = new MainCourse("Steak", "Rp.120.000", 4.5, "Main Course", "Yes");
- MainCourse mainCourse2 = new MainCourse("Spaghetti Bolognese", "Rp.55.000", 3.4, "Main Course", "No");
- MainCourse mainCourse3 = new MainCourse("Chicken Parmesan", "Rp.45.000", 4.4, "Main Course", "Yes");
- MainCourse mainCourse4 = new MainCourse("Grilled Salmon", "Rp.85.000", 3.6, "Main Course", "No");
- MainCourse mainCourse5 = new MainCourse("Ribs", "Rp.100.000", 4.0, "Main Course", "No");

- MainCourse mainCourse6 = new MainCourse("Pizza", "Rp.65.000", 3.4, "Main Course", "No");
- MainCourse mainCourse7 = new MainCourse("Burger", "Rp.50.000", 5.0, "Main Course", "Yes");

Object ini merupakan nama nama dan informasi menu khusus MainCourse. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama menu yang ada adalah Steak, Spaghetti Bolognese, Chicken Parmesan, Grilled Salmon, Ribs, Pizza, Burger dengan harga & rating yang bervariasi.

- Dessert dessert1 = new Dessert("Chocolate Cake", "Rp.28.000", 4.8, "Dessert", "Yes");
- Dessert dessert2 = new Dessert("Tiramisu", "Rp.25.000", 3.9, "Dessert", "No");
- Dessert dessert3 = new Dessert("Cheesecake", "Rp.30.000", 4.3, "Dessert", "Yes");
- Dessert dessert4 = new Dessert("Ice Cream Sundae", "Rp.18.000", 4.7, "Dessert", "Yes");

Object ini merupakan nama nama dan informasi menu khusus Dessert. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama menu yang ada adalah Chocolate Cake, Tiramisu, Cheesecake, dan Ice Cream Sundae dengan harga & rating yang bervariasi.

- Drinks drink1 = new Drinks("Orange Juice", "Rp.10.000", 4.5, "Cold", "Yes");
- Drinks drink2 = new Drinks("Apple Juice", "Rp.10.000", 3.9, "Cold", "No");
- Drinks drink3 = new Drinks("Avocado Juice", "Rp.10.000", 3.8, "Cold", "No");
- Drinks drink4 = new Drinks("Iced Tea", "Rp.5.000", 4.5, "Cold", "Yes");
- Drinks drink5 = new Drinks("Coffee", "Rp.7.000", 4.0, "Hot", "No");

Object ini merupakan nama nama dan informasi menu khusus Drink. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama minuman yang ada adalah Orange Juice, Apple Juice, Avocado Juice, Iced Tea, dan Coffee dengan harga & rating yang bervariasi.

- MenuStock menuStock = new MenuStock();

Object ini digunakan untuk mendeklarasikan MenuStock, yaitu jumlah stock setiap makanan.

- Koki koki 1 = new Koki("Ilham", "Head Chef", "Appetizer", 5, true, menuStock);
- Koki koki2 = new Koki("Juna Agus", "Head Chef", "Main Course", 13, true, menuStock);
- Koki koki 3 = new Koki("Renatta", "Pastry Chef", "Dessert", 10, true, menuStock);
- Koki koki 4 = new Koki("Akbar", "Sous Chef", "Drink", 8, true, menuStock);

Object ini merupakan nama dan informasi dari para chef, object ini berisikan Nama chef, Position, Speciality, Experience, Dan apakah mempunyai sertifikat (terpercaya) atau tidak yang dimana menggunakan boolean

- ShiftKoki Skoki1 = new ShiftKoki("Ilham", "Head Chef", "Appetizer", 5, true, menuStock, "08:00 - 10:00");
- ShiftKoki Skoki2 = new ShiftKoki("Juna Agus", "Head Chef", "Main Course", 13, true, menuStock, "10:00 - 13:00");
- ShiftKoki Skoki3 = new ShiftKoki("Renatta", "Pastry Chef", "Dessert", 10, true, menuStock, "13:00 - 16:00");
- ShiftKoki Skoki4 = new ShiftKoki("Akbar", "Sous Chef", "Drink", 8, true, menuStock, "16:00 - 18:00");

Object ini merupakan turunan dari kelas Koki. Object ini berfungsi untuk mendeklarasikan shift dan atribut koki yang ada.

- Restaurant restaurant = new Restaurant("Five Spices", "PIK", 1945, "Robben Van Deuc Wewengkang & Setiadi Deur Vand Teach", 4.89);

Object ini berfungsi untuk mendeklarasikan informasi restoran Five Spices. Mengandung informasi seperti nama restoran, lokasi restoran, dibuat sejak tahun berapa, nama owner restoran dan rating restoran.

- `FiveSpicesQueue queue = new FiveSpicesQueue();`

Objek ini digunakan untuk mendeklarasikan antrian yang ada di dalam restoran

b. Attributes:

Atribut yang dipakai dalam proses transaksi restoran kami merupakan ; String nama, Double harga, dan int Jumlah. Atribut ini digunakan di dalam kelas pesanan, yang dimana nanti akan disimpan di dalam ArrayList pesanan yang ada. String nama berfungsi sebagai atribut nama makanan ataupun minuman. Double Harga berfungsi sebagai atribut harga makanan yang diinput. Dan int jumlah merupakan atribut jumlah makanan yang dipesan oleh user.

c. Methods:

Method yang digunakan di dalam code kami untuk melakukan proses transaksi adalah `hitungTotal(ArrayList<Pesanan> pesananList)` yang berfungsi untuk menghitung semua total makanan yang dipesan dengan mengalikan jumlah dengan harga makanan. Setelah itu ada method yang bernama `cetakStruk(ArrayList<Pesanan> pesananList)` yang berfungsi untuk mencetak struk yang berisikan semua makanan / minuman yang dipesan. Juga ada method `handlePayment(double amount, ArrayList<Pesanan> pesananList)` yang berfungsi untuk menangani proses pembayarn. Di dalam restoran kami, kita bisa membayar secara cash maupun debit card untuk customer agar customer dapat membayar makanan yang sudah mereka pesan.

d. Process Transactions:

Transaksi dalam sistem restoran dilakukan secara sistematis agar mencegah adanya kesalahan di dalam pemesanan makanan ataupun minuman.

- Pelanggan tiba dan melihat menu - menu yang diberikan.
- Pelanggan melakukan pemesanan (objek Pesanan dibuat, yang berisi item dari Menu berisikan nama makanan / minuman, harga dan jumlah).
- Pelanggan memilih meja makan untuk tempat mereka makan, pelanggan juga bisa mengcancel meja yang mereka ingin duduki. Setelah itu pelanggan mengambil nomor antrian yang ada dan mengkonfirmasi apakah pelanggan tersebut VIP atau tidak.
- Pesanan dikirim ke dapur (Koki memproses pesanan dengan interface Iprepare).
- Pesanan disiapkan oleh koki menggunakan interface Iservefood dan dikirim ke pelanggan.
- Pelanggan membayar pesanan yang sudah mereka pesan (cash / debit).

3. Question 3: Sub-CLO 3, Weight(56.24%)

In the Final Project report, explain how to apply the following points:

- a. The structure of the linked list chosen in the project (single, double, circular), and explain the reasons for using that type of linked list.
- b. Must utilize one of the Stack or Queue data structures on at least one of the objects, and explain the reasons for using Stack or Queue on that object
- c. Explain the implementation of the object in the selected study case in the data structure used

a. Answer:

Single Linked List merupakan struktur data fundamental yang digunakan dalam Computer Science untuk mengorganisasi elemen yang biasa disebut node terhubung satu sama lain menggunakan referensi ataupun link pada linear. Setiap node memiliki dua bagian utama yaitu data terdiri atas nilai dan pointer menunjuk ke node berikutnya dalam linked list. Terdiri atas dua bagian, bagian depan disebut head dan bagian belakang disebut tail.

Kegunaan Linked List yaitu ukurannya dinamis yang mudah kurang dan tambah tanpa alokasi memori besar dari awal. Kemudian ada insert dan delete data yang efisien karena elemen ini hanya perlu disambungkan ulang dan tidak ada batasan ukuran yang tetap seperti array. Lalu mengimplementasikan struktur data meliputi stack, queue. Terdapat juga kelemahan karena akses acak tidak efisien dan overhead penyimpanan tambahan untuk setiap tautan sehingga didasarkan kebutuhan spesifik dari masalah yang dihadapi. Operasinya terdiri atas traversal untuk mengetahui jumlah data yang kita punya di linked list, insertion menambah sebuah node depan, tengah dan belakang serta stack mencari data sesuai kriteria.

```
// Terdapat method untuk menampilkan informasi dari koki
public void displayKokiInfo (){
    System.out.println("Name : " + getName());
    System.out.println("Position : " + getPosition());
    System.out.println("Speciality : " + getSpeciality());
    System.out.println("Experience : " + getExperience());
    System.out.println("Certified Chef : " + isCertified());
    System.out.println(x:"");
}
```

```
private void addToLinkedList(Koki koki) {
    // Jika head masih null, maka koki baru ini menjadi head dari linked list
    if (head == null) {
        head = koki;
        // Jika head sudah ada, maka kita mulai dari head
    } else {
        Koki current = head;
        // Iterasi hingga menemukan node terakhir (next == null)
        while (current.next != null) {
            current = current.next;
        }
        // Menambahkan koki baru di akhir linked list
        current.next = koki;
    }
}
```

```
// SINGLE LINKED LIST MEMAKAI NEXT.
public static void displayAllKokiInfo() {
    System.out.println(x:"-----");
    System.out.println(x: " Our Chefs ");
    System.out.println(x:"-----");
    Koki current = head;
    while (current != null) {
        current.displayKokiInfo();
        current = current.next;
    }
}
```

Dalam Code kami, kami memakai Singly Linked List ditunjukkan lewat method “addToLinkedList(Koki koki) dengan menambahkan data ke node baru ‘koki’ ke akhir single linked list dan “displayAllKokiInfo()” untuk melakukan iterasi dan menampilkan informasi dari setiap node dalam single linked list. Dimana dalam konteks ini ditunjukkan head sebagai node pertama dan mereka memiliki referensi ke node berikutnya serta proses iterasi dilakukan hingga mencapai node terakhir.

Sumber: [Singly Linked List Tutorial - GeeksforGeeks](#)

Doubly Linked List merupakan pengembangan dari Single Linked List dimana linknya ada dua bagian yaitu next dan previous. Jadinya head berarti previous null dan tail menuju null merupakan data terakhir. Terdapat juga push untuk insert front dan append untuk insert back. Kegunaan Doubly Linked List yaitu pengimplementasiannya digunakan sebagai dasar struktur data lain seperti stack, queue dan dequeue. Lalu penambahan ataupun penghapusan elemen yang mudah tanpa memerlukan alokasi memori besar dari awal serta tidak ada batasan ukuran tetap seperti array.

```
// Referensi menuju ke node pertama dalam daftar terhubung (log paling baru)
private LogNode head;

// Referensi menuju ke node terakhir dalam daftar terhubung (log paling lama)
private LogNode tail;

// Referensi menuju ke node yang sedang dipilih saat ini
private LogNode current;

public void addLog(int day, int customers, double revenue) {
    LogNode newNode = new LogNode(day, customers, revenue);
    if (head == null) {
        head = newNode;
        tail = newNode;
    } else {
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }
}

public void printCurrentLog() {
    if (current != null) {
        System.out.println("Log" + (current.day == 0 ? " Hari ini" : " - " + current.day + " hari yang lalu"));
        System.out.println("Jumlah pelanggan : " + current.customers);
        System.out.println("Pendapatan: Rp " + current.revenue);
        System.out.println("====");
    } else {
        System.out.println("Log kosong.");
    }
}
```

Dalam Code kami, kami memakai Doubly Linked List di dalam class ActivityLog ditunjukkan oleh deklarasi kelas Node dimana menggunakan referensi ‘next’ ditujukan pada node selanjutnya dan ‘prev’ untuk node sebelumnya. Kelas AvtivityLog merupakan class yang berfungsi untuk menyimpan data Revenue Restoran untuk 7 hari kebelakang (History). Menjadikan Owner restoran dapat melihat revenue restoran untuk 7 hari ke belakang. Kemudian mendeklarasikan atribut ‘LogNode’ yang sama halnya seperti tadi terdiri dari ‘head’, ‘tail’ serta ‘current’ untuk melacak node pertama, terakhir dan node saat ini. Terdapat method ‘addLog’ yang menambahkan node baru di awal list dan mengatur referensi ‘next’ dan ‘prev’ untuk menjaga hubungan dua arah dan juga method ‘printCurrentLog’ untuk menampilkan informasi dari node yang saat ini dipilih.

Sumber: [Introduction to Doubly Linked List – Data Structure and Algorithm Tutorials - GeeksforGeeks](#)

Single Circular Linked List merupakan struktur data linear terdiri dari sekelompok simpul yang terhubung secara siklik maksudnya simpul terakhir yang ada pada linked list terhubung kembali ke simpul pertama membentuk sebuah lingkaran. Jadi, setiap simpul memiliki dua bagian yaitu data dan tautan ke simpul berikutnya dalam urutan. Kegunaan Circular Linked List yaitu mengelola data dalam bentuk lingkaran, mengalokasikan memori dengan cara efisien karena dapat mengelola daftar blok memori yang tersedia dan terpakai serta mengimplementasikan struktur data terhubung berulang.

```
// Terdapat method untuk menampilkan nama, posisi, keahlian, dan waktu shift dari setiap Koki
public void viewShiftKoki() {
    System.out.println(x:"-----");
    System.out.println(x: "      Koki Shifts      ");
    System.out.println(x:"-----");
    // Menampilkan informasi tentang koki
    System.out.println("Name: " + getName()); // Menampilkan nama
    System.out.println("Position: " + getPosition()); // Menampilkan posisi
    System.out.println("Speciality: " + getSpeciality()); // Menampilkan keahlian
    System.out.println("Shift Time: " + getShiftWaktu()); // Menampilkan waktu shift
    System.out.println(x: "_");
}
```

```
public class ShiftKoki extends Koki {
    // Terdiri atas atribut privat untuk menyimpan waktu shift
    private String shiftWaktu;

    // Terdapat konstruktor untuk menginisialisasi objek ShiftKoki
    public ShiftKoki(String name, String position, String speciality, int experience, boolean isCertified,
                      MenuStock menuStock, String shiftWaktu) {
        // Memanggil constructor superclass (Koki)
        super(name, position, speciality, experience, isCertified, menuStock);
        // Menginisialisasi atribut shiftWaktu
        this.shiftWaktu = shiftWaktu;
    }
}
```

```
public class ShiftSCLList {
    // Terdiri atas node privat saat ini
    private Node current;

    // Terdapat kelas privat Node untuk single circular linked list
    private class Node {
        // Terdiri atas data berupa objek ShiftKoki
        ShiftKoki data;
        // Terdiri referensi ke node selanjutnya
        Node next;

        // Terdapat konstruktor untuk Node
        public Node(ShiftKoki data) {
            this.data = data;
            this.next = null;
        }
    }
}
```

```

// Terdapat method untuk menambahkan shift ke dalam list
public void addShift(ShiftKoki shift) {
    Node newNode = new Node(shift);
    if (isEmpty()) {
        head = newNode;
        newNode.next = head;
        // Mengatur current ke head yang baru ditambahkan
        current = head;
    } else {
        Node temp = head;
        while (temp.next != head) {
            temp = temp.next;
        }
        temp.next = newNode;
        newNode.next = head;
    }
    size++;
}

// Terdapat method untuk menampilkan informasi shift saat ini
public void displayCurrentShift() {
    // Menampilkan informasi shift koki
    if (current != null) {
        current.data.viewShiftKoki();
    } else {
        System.out.println("Shift not available.");
    }
}

```

Dalam Code kami, kami memakai Class turunan dari Koki yang bernama ShiftKoki. Kelas ShiftKoki memiliki atribut baru yaitu String shiftWaktu. Kami membuat method addShift untuk memasukan data shift koki yang akan dimasukan ke Single Circular Linked List. Kami menggunakan Single Circular Linked List yang berfungsi untuk mencetak informasi Shift koki yang ada. Dengan menggunakan fitur dari Single Circular Linked List, customer sebagai user bisa melihat shift yang ada dan menekan next untuk melihat shift selanjutnya.

Sumber: [Circular Linked List Implementation in Java - GeeksforGeeks](#)

Circular Double Linked List merupakan struktur data linear terdiri dari sekelompok simpul membentuk lingkaran dan tiap simpul memiliki navigasi dua link yaitu previous dan next untuk ke simpul sebelumnya dan berikutnya disesuaikan dengan arah panahnya. Kegunaan Circular Doubly Linked List yaitu memungkinkan untuk traversing maju dan mundur secara efisien maksudnya menelusuri node sampai selesai dan setiap link harus terhubung serta meningkatkan efisiensi dalam penggunaan cache karena data yang sering diakses memungkinkan akses cepat ke data baik dari depan maupun belakang.

```
private static class MenuItem {
    Menu menu;
    MenuItem next;
    MenuItem prev;

    MenuItem(Menu menu) {
        this.menu = menu;
        this.next = null;
        this.prev = null;
    }
}
```

```
// Mendeklarasikan kelas MenuCDLList
public class MenuCDLList {
    private MenuItem head;
    private MenuItem current;

    // Terdapat method untuk menambahkan menu ke dalam linked list
    public void addMenu(MenuItem menuItem) {
        MenuItem newNode = new MenuItem(menuItem);
        if (head == null) {
            head = newNode;
            head.next = head;
            head.prev = head;
            current = head;
        } else {
            MenuItem last = head.prev;
            last.next = newNode;
            newNode.prev = last;
            newNode.next = head;
            head.prev = newNode;
        }
    }
}
```

```
// Terdapat method untuk menampilkan menu saat ini
public void displayCurrentMenu() {
    if (current != null) {
        System.out.println("=====");
        System.out.println("          Current Menu");
        System.out.println("=====");
        current.menu.displayMenu();
        System.out.println("-----");
    } else {
        System.out.println("Menu not available.");
    }
}
```

```
// Menambahkan menu ke daftar menu
// Appetizer
menuListAp.addMenu(appettizer1);
menuListAp.addMenu(appettizer2);
menuListAp.addMenu(appettizer3);
menuListAp.addMenu(appettizer4);
menuListAp.addMenu(appettizer5);
```

Dalam Code kami, kami menggunakan Circular Double Linked List ditunjukkan oleh deklarasi kelas MenuCDLList yang mendefenisikan node dari kelas MenuItem yang menginisialisasi atribut ‘menu’ yang menyimpan objek Menu mewakili item menu, ‘next’ dan ‘prev’ diatur ke referensi this menandakan node pertama dan terakhir dalam Linked List terhubung satu sama lain. Menjadikan Linked list dapat menunjuk ke node next, prev dan mengulang ke node pertamanya kembali. Di dalam code kami, kami menggunakan Circular Double Linked List untuk mencetak menu menu yang ada di dalam code kami. Menu yang nanti dipilih dengan user juga bervariasi tergantung dengan tipe menu, yaitu Appetizer, MainCourse, Dessert, Drinks. User nanti dapat mencetak menu yang dipilih dan bisa melihat menu selanjutnya dengan menekan “n” dan sebelumnya dengan menekan “p”.

Sumber: [Circular Doubly Linked List - javatpoint](#)

b. Answer:

Stack ataupun tumpukan merupakan bagian dari linked list dan merupakan struktur data yang mengikuti prinsip LIFO (Last In, First Out) maksudnya elemen terakhir dimasukkan ke dalam stack menjadi elemen pertama yang diambil dari stack. Pengoperasiannya sendiri terdiri dari Push untuk menambah elemen kedalam Stack, Pop untuk menyingkirkan elemen teratas dari Stack dan menjadikan yang paling bawah sebagai head. Dilanjutkan dengan Peek / Top dengan melihat elemen paling atas serta isEmpty mengecek stack kosong atau tidak.

Kegunaan Stack yaitu mengelola data dan fungsi secara efisien sama halnya dengan pengontrolan memori dialokasikan atau tidak. Ia memungkinkan dalam penambahan dan pengurangan elemen dengan urutan yang telah ditentukan lewat pengekspresian postfix atau

infix dan digunakan pada berbagai aplikasi untuk manajemen data, fungsi dan memori yang efisien.

```
// Terdapat konstruktor untuk membuat objek Restaurant dengan parameter yang diberikan
public Restaurant(String name, String location, int year, String owner, double rating) {
    // Menginisiasi atribut-atribut dari objek Restaurant sesuai dengan parameter yang diberikan
    this.name = name;
    this.location = location;
    this.year = year;
    this.owner = owner;
    this.rating = rating;
    this.status = "Available"; // Set status awal restoran sebagai "Available"
    this.feedbackStack = new Feedback(); // Inisialisasi feedbackStack
}
```

```
// Mendeklarasikan kelas Feedback
class Feedback {
    // Terdiri atas kelas privat static Node
    private static class Node {
        // Terdiri atas atribut untuk menyimpan feedback dan referensi ke node berikutnya
        String feedback;
        Node next;

        // Terdapat konstruktor Node yang menerima feedback
        Node(String feedback) {
            this.feedback = feedback;
        }
    }

    // Terdapat atribut privat untuk menunjuk ke node teratas dari stack
    private Node top;

    // Terdapat konstruktor Feedback menginisialisasi top sebagai null
    public Feedback() {
        top = null;
    }
}
```

```
// Terdapat method untuk menampilkan semua feedback di stack
public void displayFeedbacks() {
    Node current = top;
    // Jika tidak ada feedback, tampilkan pesan bahwa belum ada feedback
    if (current == null) {
        System.out.println("Belum ada feedback.");
        return;
    }
    // Jika ada feedback, tampilkan semua
    System.out.println("Feedback :");
    while (current != null) {
        System.out.println("-----");
        System.out.println("++ " + current.feedback);
        System.out.println("-----\n");
        current = current.next;
    }
}
```

```

// Meminta pelanggan memasukkan feedback ke restoran
switch (infoChoice) {
    case 1:
        System.out.print(s:"Masukkan feedback Anda: ");
        String feedback = scan.nextLine();
        restaurant.addFeedback(feedback);
        break;
    case 2:
        boolean feedbackMenuQuit = false;
        while (!feedbackMenuQuit) {
            restaurant.displayFeedbacks();
            System.out.println(x:"1. Quit");
            System.out.print(s:"Pilih aksi (1): ");
            int feedbackChoice = scan.nextInt();
            if (feedbackChoice == 1) {
                feedbackMenuQuit = true;
            } else {
                System.out.println(x:"Pilihan tidak valid. Silakan coba lagi.");
            }
        }
        break;
}

```

Dalam Code kami, kami menggunakan Stack ditunjukkan dari deklarasi kelas Feedback yang berada di dalam Class Restaurant. Kelas Feedback mendefenisikan node dalam stack dan setiap node memiliki dua atribut yaitu ‘String feedback’ yang menyimpan feedback dari pelanggan dan ‘next’ merupakan referensi ke node berikutnya dalam stack dan konstruktor Feedback Class Feedback yang berada di dalam Restaurant (Inner Class). Terdapat juga method ‘displayFeedbacks()’ yang mengiterasi stack dan mencatat setiap feedback. Dengan adanya stack, feedback restoran yang dimasukan pertama akan berada menjadi feedback yang paling bawah.

Sumber: [Java Stack - Javatpoint](#)

Queue adalah struktur data yang digunakan untuk menyimpan berbagai elemen yang disusun sesuai prinsip, First In, First Out atau disingkat dari FIFO. Artinya, elemen yang masuk pertama kali pada antrian juga dikeluarkan pertama kali dari antrian. Queue mengikuti prinsip FIFO, yang berarti elemen pertama yang dimasukkan ke dalam queue akan menjadi elemen pertama yang keluar. Hal ini sering kali sangat penting dalam pemrosesan data berurutan, seperti antrian pesan, penjadwalan tugas, atau proses yang membutuhkan penanganan dalam urutan kedatangan.

Dalam OOP, queue dapat diimplementasikan dengan mudah menggunakan struktur data sederhana seperti array atau linked list. Hal ini membuatnya mudah dipahami dan diterapkan dalam berbagai konteks pemrograman.

```
class Customer {
    String nameCust;
    int nomorAntrian;
    boolean isVip;

    Customer(String nameCust, int nomorAntrian, boolean isVip) {
        this.nameCust = nameCust;
        this.nomorAntrian = nomorAntrian;
        this.isVip = isVip;
    }

    @Override
    public String toString() {
        return String.format("%-10s | %-3d | %-3s |", nameCust, nomorAntrian, isVip ? "Yes" : "No");
    }
}
```

```
public class FiveSpicesQueue {
    private Node front;
    private Node rear;
    /*
    Konstruktor FiveSpicesQueue berperan dalam
    objek antrian dibuat dengan menginisialisasi
    pada awalnya, antrian tersebut kosong */
    public FiveSpicesQueue() {
        this.front = null;
        this.rear = null;
    }
}
```

```
public void displayQueue() {
    Node temp = front;
    while (temp != null) {
        System.out.println(temp.customer);
        temp = temp.next;
    }
}
```

Dalam code, kami menggunakan Queue dimana ditunjukkan oleh deklarasi dari kelas Customer yang memiliki atribut String nameCust, int nomorAntrian, boolean isVip dan juga kelas FiveSpicesQueue yang mengatur antrian berisi Customer yang memiliki dua atribut privat yaitu ada ‘front’ dan ‘rear’ merujuk pada node pertama dan terakhir dalam antrian. Konstruktor FiveSpicesQueue menginisialisasi kedua atribut tersebut null bahwa antrian kosong. Kemudian terdapat method Enqueue yang menambahkan customer baru ke antrian diikuti method Dequeue yang menghapus customer pertama dari antrian. Terdapat kedua method yaitu ada isEmpty untuk memeriksa apakah antrian kosong atau tidak dan displayQueue yang mencetak isi antrian.

Sumber: [Queue Data Structure - GeeksforGeeks](#)

c. Answer:

Dibawah ini adalah kode yang menampilkan berbagai objek dan struktur data yang digunakan untuk memodelkan sistem manajemen restoran. Untuk detail nya tentang bagaimana objek-objek ini berinteraksi dengan struktur data yang dipilih:

- a. Restaurant:
 - Objek: Mewakili instance restoran dengan atribut seperti lokasi, nama, pemilik, dll.
 - Struktur Data: Tidak disebutkan secara eksplisit, tetapi kemungkinan merupakan kelas sederhana yang memiliki atribut ini.
- b. Feedback (Inner Class of Restaurant):
 - Objek: Mewakili satu entri umpan balik untuk restoran.
 - Struktur Data: Kelas node dalam Stack. Setiap objek Node menyimpan teks umpan balik dan referensi ke node berikutnya dalam tumpukan. Atribut feedbackStack di kelas Restoran kemungkinan mengarah ke bagian atas tumpukan ini, memungkinkan akses LIFO (Last-In-First-Out) ke entri umpan balik.
- c. Menu:
 - Objek: Mewakili satu item menu dengan rincian seperti nama, harga, peringkat, jenis, dan status favorit.
 - Struktur Data: Tidak disebutkan secara eksplisit, tetapi kemungkinan merupakan kelas sederhana yang memiliki atribut ini.
- d. MenuCDLLList (Doubly Circular Linked List):
 - Objek: Mewakili kumpulan semua item menu.
 - Struktur Data: Daftar Tertaut Melingkar Ganda yang diimplementasikan menggunakan kelas MenuItem. Setiap objek MenuItem menyimpan objek Menu dan referensi ke item berikutnya dan sebelumnya dalam daftar, memungkinkan navigasi yang efisien melalui kumpulan menu di kedua arah. Atribut head dan current di MenuCDLLList kemungkinan mengarah ke node tertentu dalam daftar.
- e. MenuStock:
 - Objek: Mewakili informasi stok terkini untuk item menu.

- Struktur Data: HashMap yang menggunakan nama item menu (String) sebagai kunci dan jumlah stok yang sesuai (Bilangan Bulat) sebagai nilai. Hal ini memungkinkan pengambilan informasi stok secara efisien untuk setiap item menu tertentu.
- f. Koki (Chef) (Inner Class):
 - Objek: Mewakili koki yang bekerja di restoran dengan detail seperti nama, posisi, spesialisasi, status sertifikasi, pengalaman, dan akses ke informasi stok menu.
 - Struktur Data: Tidak disebutkan secara eksplisit, tetapi kemungkinan merupakan kelas sederhana yang memiliki atribut ini.
- g. KokiShift (Inner Class):
 - Objek: Mewakili shift kerja koki dengan detail yang mirip dengan Koki namun berpotensi menyertakan informasi tambahan seperti waktu shift tertentu.
 - Struktur Data: Tidak disebutkan secara eksplisit, tetapi kemungkinan merupakan kelas sederhana yang memiliki atribut ini.
- h. ShiftSCLList (Singly Circular Linked List):
 - Objek: Berpotensi mewakili kumpulan shift kerja chef. Penggunaan ini kurang jelas dari cuplikan yang disediakan.
 - Struktur Data: Daftar Tertaut Melingkar Tunggal yang diimplementasikan menggunakan kelas Node. Setiap objek Node kemungkinan memiliki objek KokiShift dan referensi ke node berikutnya dalam daftar, sehingga memungkinkan iterasi yang efisien melalui daftar shift.
- i. ActivityLog:
 - Objek: Mewakili entri log tunggal untuk aktivitas sehari-hari di restoran, termasuk data seperti hari, jumlah pelanggan, pendapatan yang diperoleh, dan kemungkinan referensi ke entri log sebelumnya dan berikutnya.
 - Struktur Data: Tidak disebutkan secara eksplisit, tetapi kemungkinan merupakan kelas sederhana yang memiliki atribut ini. Ini juga dapat diterapkan sebagai daftar tertaut jika menjaga urutan kronologis log itu penting.
- j. Customer:
 - Objek: Mewakili pelanggan yang mengunjungi restoran dengan detail seperti nama, nomor antrian, dan status VIP.

- Struktur Data: Tidak disebutkan secara eksplisit, tetapi kemungkinan merupakan kelas sederhana yang memiliki atribut ini.
- k. FiveSpicesQueue:
- Objek: Mewakili antrian (kemungkinan disebut "Antrian Lima Bumbu") yang digunakan untuk mengelola antrean tunggu pelanggan.
 - Struktur Data: Antrian diimplementasikan menggunakan kelas Node. Setiap objek Node kemungkinan menyimpan objek Pelanggan dan referensi ke pelanggan berikutnya dalam antrian. Atribut depan dan belakang di FiveSpicesQueue kemungkinan masing-masing menunjuk ke pelanggan pertama dan terakhir dalam antrean.
- l. Table:
- Objek: Mewakili meja makan di restoran dengan detail seperti nomor meja, status (terisi/tersedia), kemungkinan daftar semua meja (untuk mengelola ketersediaan meja), dan antrian untuk mengelola pelanggan yang menunggu untuk meja tertentu.
 - Struktur Data: Objek tabel itu sendiri kemungkinan besar menyimpan atributnya.
 - Atribut tabel di kelas lain (tidak jelas dari cuplikan) mungkin berupa ArrayList untuk menyimpan referensi ke semua objek Tabel, memungkinkan pencarian dan pengelolaan semua tabel secara efisien.
 - Atribut antrian dalam Tabel kemungkinan adalah Antrian yang diimplementasikan menggunakan struktur Node serupa.

Mata Kuliah: Object Oriented Programming - CL

Final Project Assignment, Final Exam Theory and Final Exam Lab scores will be integrated. Pay attention to the scoring rubric to get the best score.

1. Question 1: Sub-CLO 1, Weight (13.76%)

In the Final Project Project, make sure the code contain these following points:

- a. Clear object creation: attributes, constructor, getter/setter, methods
- b. Comments/additional information for each class/object and methods
- c. Proper class/object naming

Answer :

Proyek akhir kami adalah tentang restoran yang bernama Five Spices. Nama dari restoran ini memiliki arti bahwa kami menggunakan campuran bumbu rempah yang beragam. Kami memilih tema restoran untuk proyek ini karena ingin mengoptimalkan efisiensi sistem dalam berbagai aspek manajemen restoran, seperti manajemen stok, manajemen data, dan informasi penjualan, sehingga operasional berjalan lancar dan pengelolaan transaksi menjadi lebih baik, akurat, dan efisien. Dengan menggunakan Object-Oriented Programming (OOP), kami dapat menciptakan sistem manajemen restoran yang lebih reusable dan dapat digunakan kembali dalam kelas-kelas lain, seperti manajemen pesanan dan stok, sehingga mengurangi pengulangan kode. Pendekatan ini memungkinkan fokus pada bagian-bagian tertentu dari sistem tanpa perlu memodifikasi seluruh kode, yang pada gilirannya meningkatkan efisiensi pengembangan dan pemeliharaan sistem.

OOP memungkinkan representasi dunia nyata yang lebih akurat dari objek-objek yang ada dalam restoran, seperti Pelanggan, Menu Item, Pesanan, Karyawan, Stok Makanan, Koki, dan lain-lain. Dalam OOP, kita dapat membuat kelas yang mewakili objek-objek ini sehingga kode menjadi lebih mudah dipelihara. Selain itu, OOP menyediakan enkapsulasi dan perlindungan data melalui atribut dan metode. Misalnya, objek Pesanan memiliki metode untuk menghitung total tagihan, di mana hanya metode yang diotorisasi yang dapat mengubah detail pesanan. Implementasi OOP juga mendukung pewarisan dan polimorfisme, memungkinkan kita untuk membuat kelas turunan seperti Appetizer, Main Course, Dessert, dan Drinks, serta metode 'Override' yang identik dalam kelas induk atau antarmuka. Misalnya, kita dapat menurunkan subkelas Chef dengan fungsi khusus. Pendekatan ini mengurangi duplikasi kode dan meningkatkan fleksibilitas.

Di dalam sistem restoran Five Spices ini, kami juga menerapkan berbagai konsep OOP, mulai dari Class & Object, Encapsulation, Inheritance, Interface, Abstraction, Polymorphism, hingga Java Collection Framework, Single Linked List, Doubly Linked List, Single Circular Linked List, Circular Double Linked List, Stack, dan Queue. Dengan penerapan konsep-konsep OOP ini, kami memastikan bahwa semua alur operasional restoran bekerja sesuai dengan yang diharapkan, sehingga meningkatkan efisiensi dan efektivitas sistem manajemen restoran secara keseluruhan.

Kerangka Code :



Di dalam Kerangka kode kami, akan ada beberapa kelas yaitu : Restaurant, Menu, MenuCDLLList, MenuStock, Appetizer, MainCourse, Dessert, Drinks, Iprepare, Iservefood, Koki, KokiShift, ShiftSCLList, ActivityLog, Customer, FiveSpicesQueue, Table, dan Pesanan.

Untuk Atribut di dalam Kerangka kode kami adalah :

- **Restaurant** : String location, String name, int year, String owner, double rating, String nomorMeja dan String status, Feedback feedbackStack
 - Feedback: Node top; (Inner Class Restaurant)
 - Node: String feedback, Node next; (Inner Class Feedback)
- **Menu** : String nama, String harga, double rating, String type, String isFavorite;
- **MenuCDLLList** : MenuItem head, MenuItem current;
 - MenuItem : Menu menu, MenuItem next, MenuItem prev; (Inner Class MenuCDLLList)
- **MenuStock** : HashMap<String, Integer> stock, int currentStock;
 - **Koki** : String name, String position, String speciality, boolean isCertified, int experience, MenuStock menuStock;
 - Private static Koki head, Koki next (Inner Class Koki)
- **KokiShift** : String name, String position, String speciality, int experience, boolean isCertified, MenuStock menuStock, String shiftWaktu;
- **ShiftSCLList** : Node current, Node head, int size;
 - Node : ShiftKoki data, Node next; (Inner Class ShiftSCLList)
- **ActivityLog** : int day, int customers, double revenue, LogNode next, LogNode prev;
 - LogNode : LogNode head, LogNode tail, LogNode current; (Inner Class ActivityLog)
- **Customer** : String nameCust, int nomorAntrian, boolean isVip;
- **FiveSpicesQueue** : Node front, Node rear
 - Node : Customer customer, Node next; (Inner Class FiveSpicesQueue)
- **Table** : String nomorMeja, String status, ArrayList<Table> tables, Queue<String> antrian, int nomorAntrian

- **Pesanan** : String nama, double harga, int jumlah;

Konstruktor :

Konstruktor adalah metode khusus yang digunakan untuk menginisialisasi objek saat dibuat. Dalam kode di atas, terdapat beberapa konstruktor dalam berbagai kelas. Mari kita lihat berbagai macam konstruktor yang ada dan memberikan penjelasan untuk masing-masing:

Kelas	Parameter Konstruktor	Deskripsi Parameter
Restaurant	String name, String location, int year, String owner, double rating	Berisikan Informasi Restoran seperti Nama, lokasi, tahun, pemilik, rating restoran
Feedback	-	-
Node	String feedback	Digunakan untuk menyimpan pesan feedback yang diberikan saat pembuatan objek Node.
Menu	String nama, String harga, double rating, String type, String isFavorite	Digunakan untuk menaruh informasi menu seperti Nama, harga, rating, tipe, dan apakah favorit.
Appetizer	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu Appetizer Nama, harga, rating, tipe, apakah favorit
MainCourse	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu MainCourse Nama, harga, rating, tipe, apakah favorit
Dessert	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu Dessert Nama, harga, rating, tipe, apakah favorit

Drinks	String nama, String harga, double rating, String type, String isFavorite	Class turunan dari menu digunakan untuk menampung menu Drinks Nama, harga, rating, tipe, apakah favorit
MenuItem	Menu menu	Digunakan untuk menginisialisasi objek MenuItem dengan menu yang diberikan.
MenuStock	-	-
Koki	String name, String position, String speciality, int experience, boolean isCertified, MenuStock menuStock	Berisikan identitas koki itu sendiri meliputi nama, posisi, pengalaman, apakah koki tersebut bersertifikasi atau tidak dan stok menu

KokiShift	String name, String position, String speciality, int experience, boolean isCertified, MenuStock menuStock, String shiftWaktu	Class turunan dari Koki berisikan Nama, posisi, spesialisasi, pengalaman, apakah bersertifikat, stok menu, dan waktu shift koki
Node	ShiftKoki data	Digunakan untuk menyimpan data berupa objek ShiftKoki yang diberikan saat pembuatan objek Node.
ActivityLog	-	-
LogNode	Int day, int customers, double revenue	Terdiri atas atribut hari, jumlah pelanggan dan pendapatan
Customer	String nameCust, int nomorAntrian, boolean isVip	Terdiri atas data pelanggan berisi nama pelanggan, nomor antrian yang dimiliki dan apakah pelanggan termasuk pelanggan Vip atau tidak
Node	Customer customer	Digunakan untuk menyimpan data Customer yang diberikan saat pembuatan objek Node.
Table	String nomorMeja, String status	Terdiri atas nomor meja yang pelanggan dapat begitu juga dengan status meja apakah sudah dipesan atau tidak
Pesanan	String nama, double harga, int jumlah	Terdiri atas nama pesanan dengan harga dan jumlah pesanan

2. Question 2: Sub-CLO 2, Weight(30%)

In the Final Project Assignment, make sure the code contain these following points:

- a. The implementation of the OOP concept in code has been successfully carried out.
Minimum 2 concepts applied. (Inheritance, Encapsulation, Polymorphism, Interface or Abstraction)
- b. Implementation of the Transaction class involving more than 2 classes/objects
- c. Implementation of methods that are appropriate to the case study

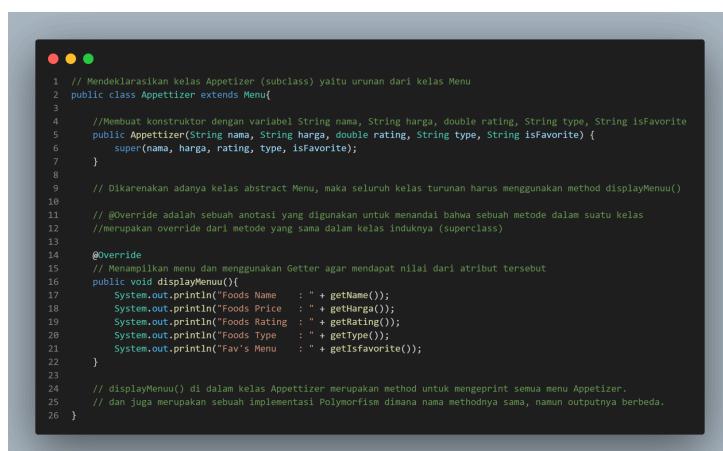
Answer :

a. Implementation of the OOP concept

Inheritance (warisan) dalam Java memungkinkan kelas untuk mewarisi metode dan atribut dari kelas lain. Kelas yang memberikan warisan disebut kelas induk atau superclass, dan kelas yang menerima warisan disebut kelas turunan atau subclass. Inheritance memungkinkan hierarki kelas yang memungkinkan untuk mengatur dan memodelkan hubungan antar kelas dalam program Java. Ini karena kelas turunan dapat menambahkan atribut dan metode tambahan atau mengganti perilaku yang ada sesuai dengan kebutuhan aplikasi. Untuk melakukan pewarisan, kata kunci ‘extends’ digunakan untuk menunjukkan kelas mana yang merupakan kelas induk. Dengan meng-extends kelas, kelas turunan dapat mewarisi karakteristik dan tingkah laku dari kelas induk. Di dalam kode kami, kami menggunakan kelas Menu sebagai Parent Class dan turunannya yaitu Appetizer, MainCourse, Dessert, dan Drinks. Dikarenakan Appetizer, MainCourse, Dessert, dan Drinks bisa dibilang adalah lanjutan atau turunan dari Menu.

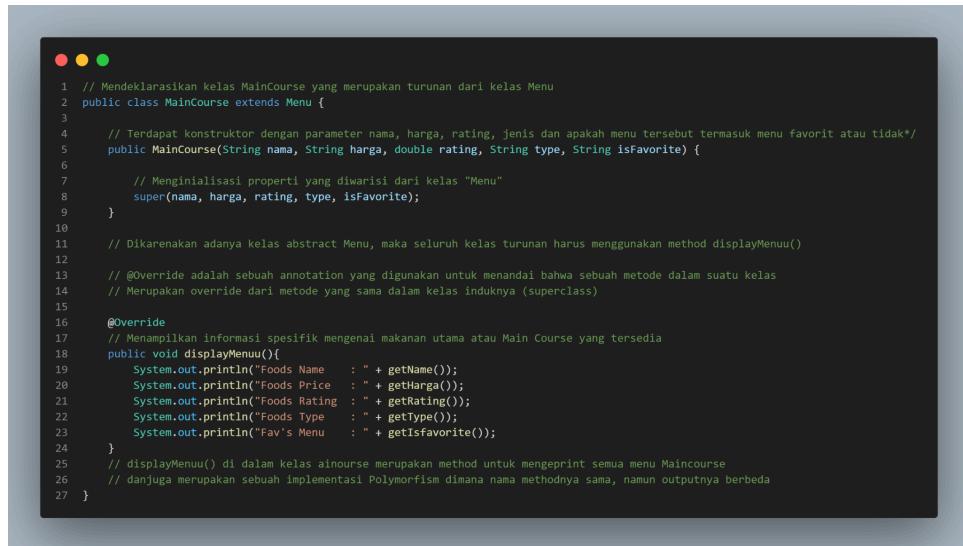
Sumber: <https://www.petanikode.com/java-oop-inheritance/>

Appetizer:



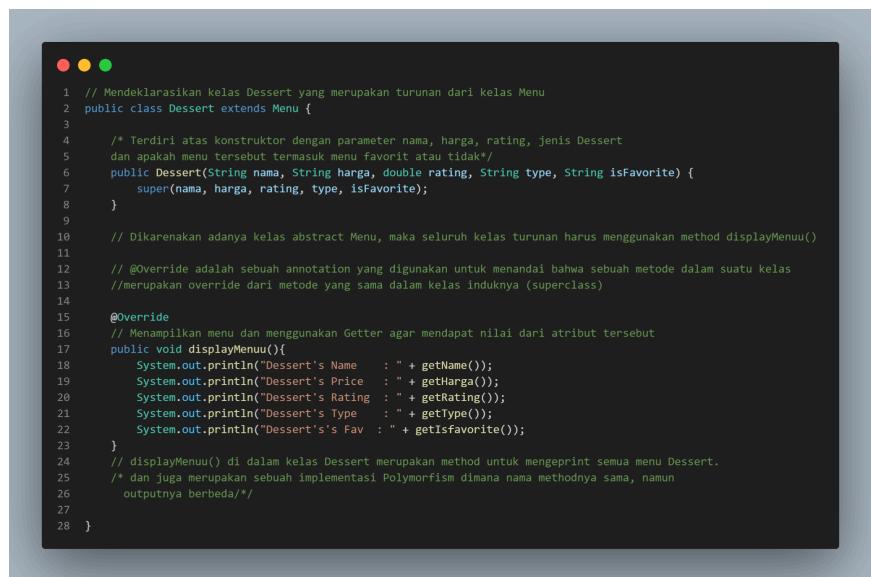
```
1 // Mendeklarasikan kelas Appetizer (subclass) yaitu turunan dari kelas Menu
2 public class Appetizer extends Menu{
3
4     // Membuat konstruktor dengan variabel String nama, String harga, double rating, String type, String isFavorite
5     public Appetizer(String nama, String harga, double rating, String type, String isFavorite) {
6         super(nama, harga, rating, type, isFavorite);
7     }
8
9     // Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
10    // @Override adalah sebuah anotasi yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
11    // merupakan override dari metode yang sama dalam kelas induknya (superclass)
12
13    @Override
14    // Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
15    public void displayMenu(){
16        System.out.println("Food's Name : " + getName());
17        System.out.println("Food's Price : " + getHarga());
18        System.out.println("Food's Rating : " + getRating());
19        System.out.println("Food's Type : " + getType());
20        System.out.println("Fav's Menu : " + getIsFavorite());
21    }
22
23
24    // displayMenu() di dalam kelas Appetizer merupakan method untuk mengeprint semua menu Appetizer.
25    // dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda.
26 }
```

MainCourse:



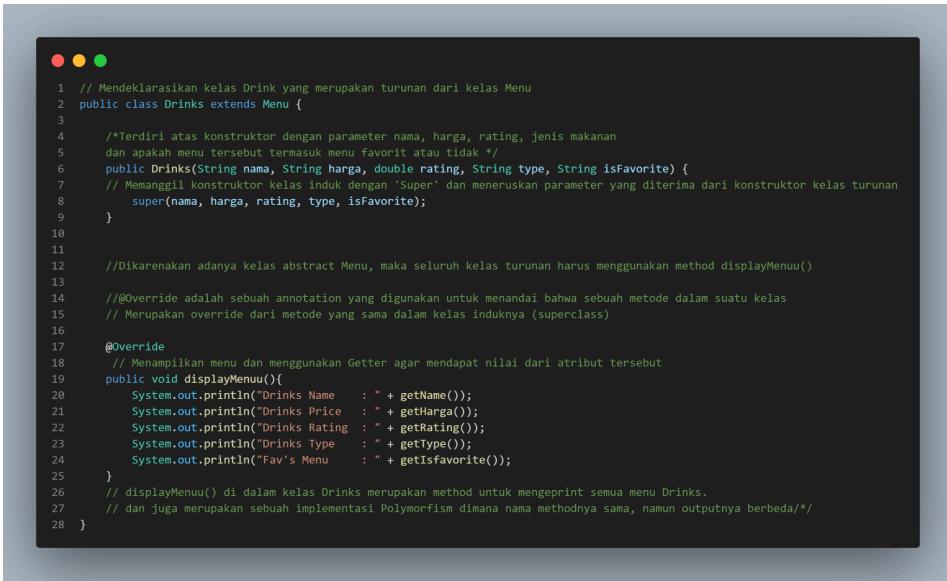
```
1 // Mendeklarasikan kelas MainCourse yang merupakan turunan dari kelas Menu
2 public class MainCourse extends Menu {
3
4     // Terdapat konstruktur dengan parameter nama, harga, rating, jenis dan apakah menu tersebut termasuk menu favorit atau tidak*
5     public MainCourse(String nama, String harga, double rating, String type, String isFavorite) {
6
7         // Menginisialisasi properti yang diwarisi dari kelas "Menu"
8         super(nama, harga, rating, type, isFavorite);
9     }
10
11    // Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
12
13    // @Override adalah sebuah annotation yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
14    // merupakan override dari metode yang sama dalam kelas induknya (superclass)
15
16    @Override
17    // Menampilkan informasi spesifik mengenai makanan utama atau Main Course yang tersedia
18    public void displayMenu(){
19        System.out.println("Foods Name : " + getName());
20        System.out.println("Foods Price : " + getHarga());
21        System.out.println("Foods Rating : " + getRating());
22        System.out.println("Foods Type : " + getType());
23        System.out.println("Fav's Menu : " + getIsfavorite());
24    }
25    // displayMenu() di dalam kelas ainourse merupakan method untuk mengeprint semua menu Maincourse
26    // dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda
27 }
```

Dessert :



```
1 // Mendeklarasikan kelas Dessert yang merupakan turunan dari kelas Menu
2 public class Dessert extends Menu {
3
4     /* Terdiri atas konstruktur dengan parameter nama, harga, rating, jenis Dessert
5      dan apakah menu tersebut termasuk menu favorit atau tidak*/
6     public Dessert(String nama, String harga, double rating, String type, String isFavorite) {
7         super(nama, harga, rating, type, isFavorite);
8     }
9
10    // Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
11
12    // @Override adalah sebuah annotation yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
13    // merupakan override dari metode yang sama dalam kelas induknya (superclass)
14
15    @Override
16    // Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
17    public void displayMenu(){
18        System.out.println("Dessert's Name : " + getName());
19        System.out.println("Dessert's Price : " + getHarga());
20        System.out.println("Dessert's Rating : " + getRating());
21        System.out.println("Dessert's Type : " + getType());
22        System.out.println("Dessert's Fav : " + getIsfavorite());
23    }
24    // displayMenu() di dalam kelas Dessert merupakan method untuk mengeprint semua menu Dessert.
25    /* dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun
26       outputnya berbeda*/
27
28 }
```

Drinks:



```
1 // Mendeklarasikan kelas Drink yang merupakan turunan dari kelas Menu
2 public class Drinks extends Menu {
3
4     /*Terdiri atas konstruktor dengan parameter nama, harga, rating, jenis makanan
5     dan apakah menu tersebut termasuk menu favorit atau tidak */
6     public Drinks(String nama, String harga, double rating, String type, String isFavorite) {
7         // Memanggil konstruktor kelas induk dengan 'Super' dan meneruskan parameter yang diterima dari konstruktor kelas turunan
8         super(nama, harga, rating, type, isFavorite);
9     }
10
11
12     //Dikarenakan adanya kelas abstract Menu, maka seluruh kelas turunan harus menggunakan method displayMenu()
13
14     //@Override adalah sebuah annotation yang digunakan untuk menandai bahwa sebuah metode dalam suatu kelas
15     // Merupakan override dari metode yang sama dalam kelas induknya (superclass)
16
17     @Override
18     // Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
19     public void displayMenu(){
20         System.out.println("Drinks Name : " + getName());
21         System.out.println("Drinks Price : " + getHarga());
22         System.out.println("Drinks Rating : " + getRating());
23         System.out.println("Drinks Type : " + getType());
24         System.out.println("Fav's Menu : " + getIsfavorite());
25     }
26     // displayMenu() di dalam kelas Drinks merupakan method untuk mengeprint semua menu Drinks.
27     // dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda/*
28 }
```

Interface: Interface adalah program yang memungkinkan pengguna untuk berinteraksi dengan perangkatnya secara langsung atau melalui jaringan. Dengan kata lain, interface adalah sekumpulan deklarasi metode tetapi tidak memiliki implementasi konkretnya. Interface memungkinkan sebuah kelas menggambarkan tugasnya tanpa mengetahui caranya.

Interface memungkinkan untuk mendefinisikan perilaku yang harus dimiliki oleh suatu objek tanpa terikat pada rincian implementasi. Ini memudahkan pembuatan kelas dengan perilaku yang sama tetapi implementasi yang berbeda. Semua kelas lain dapat mengimplementasikan interface dengan kata kunci ‘implements’. Saat sebuah kelas mengimplementasikan sebuah interface, itu harus memberikan implementasi konkret untuk semua metode yang dideklarasikan dalam interface tersebut. Di dalam kode kami, kami membuat interface Iprepare dan Idisplay. Kedua interface ini nantinya akan digunakan oleh class Koki untuk menyiapkan makanan dan menyajikan makanan kepada customer.

Sumber: [Apa Itu Interface? Pengertian, Jenis, dan Contohnya - Rumahweb](#)

Polymorphism dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “bentuk” method yang berbeda-beda meskipun namanya sama. Dapat dibilang bahwa polymorphism mengacu pada kemampuan suatu objek untuk memiliki berbagai bentuk atau perilaku. Polymorphism memungkinkan penggunaan objek dari kelas tertentu sebagai objek

dari kelas yang lebih umum, yang kemudian dapat berperilaku sesuai dengan jenis spesifiknya. Ini meningkatkan fleksibilitas penggunaan objek dan memungkinkan untuk menangani berbagai situasi dengan cara yang lebih umum. Di dalam code kami, kami menggunakan “@Override” untuk method DisplayMenuu() untuk setiap subclass, yang dimana tiap method akan menyajikan output yang berbeda.

Sumber: [Belajar Java OOP: Memahami Prinsip Polimorfisme dalam OOP \(petanikode.com\)](https://petanikode.com)

```
// Terdapat method displayMenuu() yang akan digunakan kepada subclass Menu nanti
public void displayMenuu() {

}
```

```
@Override
// Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
public void displayMenuu(){
    System.out.println("Foods Name      : " + getName());
    System.out.println("Foods Price     : " + getHarga());
    System.out.println("Foods Rating   : " + getRating());
    System.out.println("Foods Type     : " + getType());
    System.out.println("Fav's Menu      : " + getIsfavorite());
}
```

```
@Override
// Menampilkan menu dan menggunakan Getter agar mendapat nilai dari atribut tersebut
public void displayMenuu(){
    System.out.println("Drinks Name     : " + getName());
    System.out.println("Drinks Price    : " + getHarga());
    System.out.println("Drinks Rating   : " + getRating());
    System.out.println("Drinks Type    : " + getType());
    System.out.println("Fav's Menu      : " + getIsfavorite());
}
// displayMenuu() di dalam kelas Drinks merupakan method untuk mengeprint semua menu Drinks.
// dan juga merupakan sebuah implementasi Polymorfism dimana nama methodnya sama, namun outputnya berbeda/*
```

Abstraction adalah suatu prinsip dalam pemrograman OOP yang digunakan untuk menghilangkan atau menghapus karakteristik dari suatu objek agar mengurangi kompleksitasnya. Sederhananya, abstraksi memungkinkan kita untuk menemukan dan mendefinisikan perilaku objek tanpa memperhatikan implementasi khusus mereka. Dengan menggunakan abstraksi, kita dapat membuat kelas yang mewakili ide-ide tingkat tinggi yang kemudian dapat diimplementasikan dalam kelas yang lebih khusus. Ini memungkinkan untuk membuat model yang lebih sederhana dan mudah dipahami untuk mengatasi kompleksitas pengembangan perangkat lunak. Dalam Java, kelas abstrak dan interface biasanya digunakan untuk menerapkan abstraksi. Kelas abstrak adalah kelas yang tidak dapat diinstansiasi dan

memiliki metode abstrak, yang harus diterapkan oleh kelas-kelas turunannya. Di dalam kode kami, class Menu akan menjadi class abstract. Class Menu akan memaksa kelas turunannya yaitu Appetizer, MainCourse, Dessert, dan Drinks untuk melakukan method displayMenu().

Sumber: [Abstraction adalah: Pengertian dan Analogi Sederhananya | kumparan.com](https://www.kumparan.com)

```
 1 // Mendeklarasikan kelas Abstract Menu
 2 public abstract class Menu {
 3     // Terdiri atas atribut privat String nama, String harga, double rating, String type, String isFavorite
 4     private String nama;
 5     private String harga;
 6     private double rating;
 7     private String type;
 8     private String isFavorite;
 9
10    /*String nama digunakan untuk nama menu, String digunakan untuk harga menu,
11    double rating digunakan untuk rating sebuah menu dari 1-5, String type digunakan untuk mengetahui jenis menu,
12    String isFavorite digunakan untuk mengetahui apakah menu itu favorit atau bukan*/
13
14    // Terdapat konstruktor dengan variabel String nama, String harga, double rating, String type, String isFavorite
15    public Menu (String nama , String harga, double rating, String type, String isFavorite){
16        this.nama = nama ;
17        this.harga = harga;
18        this.rating = rating;
19        this.type = type;
20        this.isFavorite = isFavorite;
21    }
22
23    // Menggunakan method Getter
24    public String getName () {
25        return nama;
26    }
27
28    public String getHarga(){
29        return harga;
30    }
31
32    public double getRating (){
33        return rating;
34    }
35
36    public String getType(){
37        return type;
38    }
39
40    public String getIsfavorite (){
41        return isFavorite;
42    }
43
44    // Menggunakan method Setter
45    public void setIsfavorite (String isFavorite) {
46        this.isFavorite = isFavorite;
47    }
48
49    // Terdapat method displayMenu() yang akan digunakan kepada subclass Menu nanti
50    public void displayMenuu() {
51
52    }
53
54    // Terdapat method displayMenuWithBorders() yang akan digunakan di Main.
55    /*displayMenuWithBorders() merupakan method untuk menampilkan menu dengan
56    border agar saat user menampilkan menu, menu yang ditampilkan itu lebih rapi*/
57
58 }
```

Encapsulation adalah salah satu prinsip dasar dalam Object Oriented Programming (OOP) yang bertujuan untuk membungkus data dalam satu unit. Dalam pemrograman berorientasi objek (OOP), encapsulation adalah konsep penting yang bertujuan untuk menyatukan / membungkus data (atribut) dan fungsi (metode) yang terkait ke dalam sebuah unit tunggal yang disebut class. Dengan adanya encapsulation, detail internal implementasi objek disembunyikan dari program lain dan dunia luar, sehingga yang terlihat sekumpulan metode public yang bisa digunakan untuk berinteraksi dengan objek tersebut. Dengan adanya encapsulation, maka ada yang kita buat akan lebih aman. Dengan membatasi akses langsung ke atribut, kita bisa melindungi data dari perubahan yang tidak diinginkan. Encapsulation juga membuat kode lebih terorganisir dan modular, sehingga lebih mudah dipahami, dipelihara, dan digunakan kembali. Di dalam atribut kode kami, kami membuat semua atribut berstatus private. Dan mengaksesnya menggunakan getter setter method.

Sumber: [Apa itu Encapsulation? Pengertian dan contoh 2023 | RevoU](#)

```
// Mendeklarasikan kelas Abstract Menu
public abstract class Menu {
    // Terdiri atas atribut privat String nama, String harga, double rating, String type, String isFavorite
    private String nama;
    private String harga;
    private double rating;
    private String type;
    private String isFavorite;

    // Menggunakan method Getter
    public String getName () {
        return nama;
    }

    public String getHarga(){
        return harga;
    }

    public double getRating (){
        return rating;
    }

    public String getType(){
        return type;
    }

    public String getIsfavorite (){
        return isFavorite;
    }
}
```

b. Implementation of the transaction class

```
ArrayList<Pesanan> pesananList = new ArrayList<>();
```

Object ini berfungsi untuk menyimpan List list pesanan yang akan dipesan oleh user.

- Appetizer appetizer1 = new Appetizer("Caesar Salad", "Rp.20.000", 4.2, "Appetizer", "Yes");
- Appetizer appetizer2 = new Appetizer("Mozzarella Sticks", "Rp.17.000", 4.3, "Appetizer", "Yes");
- Appetizer appetizer3 = new Appetizer("Bruschetta", "Rp.18.000", 3.2, "Appetizer", "No");
- Appetizer appetizer4 = new Appetizer("Shrimp Cocktail", "Rp.28.000", 3.4, "Appetizer", "No");
- Appetizer appetizer5 = new Appetizer("Meat Risoles", "Rp.5.000", 4.3, "Appetizer", "Yes");

Object ini merupakan nama-nama dan informasi menu khusus Appetizer. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama menu yang ada adalah Caesar Salad, Mozzarella Sticks, Bruschetta, Shrimp Cocktail, dan Meat Risoles dengan harga & rating yang bervariasi.

- MainCourse mainCourse1 = new MainCourse("Steak", "Rp.120.000", 4.5, "Main Course", "Yes");
- MainCourse mainCourse2 = new MainCourse("Spaghetti Bolognese", "Rp.55.000", 3.4, "Main Course", "No");
- MainCourse mainCourse3 = new MainCourse("Chicken Parmesan", "Rp.45.000", 4.4, "Main Course", "Yes");
- MainCourse mainCourse4 = new MainCourse("Grilled Salmon", "Rp.85.000", 3.6, "Main Course", "No");
- MainCourse mainCourse5 = new MainCourse("Ribs", "Rp.100.000", 4.0, "Main Course", "No");
- MainCourse mainCourse6 = new MainCourse("Pizza", "Rp.65.000", 3.4, "Main Course", "No");

- MainCourse mainCourse7 = new MainCourse("Burger", "Rp.50.000", 5.0, "Main Course", "Yes");

Object ini merupakan nama nama dan informasi menu khusus MainCourse. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama menu yang ada adalah Steak, Spaghetti Bolognese, Chicken Parmesan, Grilled Salmon, Ribs, Pizza, Burger dengan harga & rating yang bervariasi.

- Dessert dessert1 = new Dessert("Chocolate Cake", "Rp.28.000", 4.8, "Dessert", "Yes");
- Dessert dessert2 = new Dessert("Tiramisu", "Rp.25.000", 3.9, "Dessert", "No");
- Dessert dessert3 = new Dessert("Cheesecake", "Rp.30.000", 4.3, "Dessert", "Yes");
- Dessert dessert4 = new Dessert("Ice Cream Sundae", "Rp.18.000", 4.7, "Dessert", "Yes");

Object ini merupakan nama nama dan informasi menu khusus Dessert. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama menu yang ada adalah Chocolate Cake, Tiramisu, Cheesecake, dan Ice Cream Sundae dengan harga & rating yang bervariasi.

- Drinks drink1 = new Drinks("Orange Juice", "Rp.10.000", 4.5, "Cold", "Yes");
- Drinks drink2 = new Drinks("Apple Juice", "Rp.10.000", 3.9, "Cold", "No");
- Drinks drink3 = new Drinks("Avocado Juice", "Rp.10.000", 3.8, "Cold", "No");
- Drinks drink4 = new Drinks("Iced Tea", "Rp.5.000", 4.5, "Cold", "Yes");
- Drinks drink5 = new Drinks("Coffee", "Rp.7.000", 4.0, "Hot", "No");

Object ini merupakan nama nama dan informasi menu khusus Drink. Object ini berisikan nama menu, harga menu, rating menu, tipe menu, dan apakah menu tersebut favorit. Nama minuman yang ada adalah Orange Juice, Apple Juice, Avocado Juice, Iced Tea, dan Coffee dengan harga & rating yang bervariasi.

Atribut yang dipakai dalam proses transaksi restoran kami merupakan ; String nama, Double harga, dan int Jumlah. Atribut ini digunakan di dalam kelas pesanan, yang

dimana nanti akan disimpan di dalam ArrayList pesanan yang ada. String nama berfungsi sebagai atribut nama makanan ataupun minuman. Double Harga berfungsi sebagai atribut harga makanan yang diinput. Dan int jumlah merupakan atribut jumlah makanan yang dipesan oleh user. Dari menu menu yang sudah ada, yang dibuat di kelas main, pengguna dapat memesan makanan dan minuman yang tersedia.

c. Implementation of the methods that are appropriate to the case study

Method yang digunakan di dalam code kami untuk melakukan proses transaksi adalah hitungTotal(ArrayList<Pesanan> pesananList) yang berfungsi untuk menghitung semua total makanan yang dipesan dengan mengalikan jumlah dengan harga makanan. Setelah itu ada method yang bernama cetakStruk(ArrayList<Pesanan> pesananList) yang berfungsi untuk mencetak struk yang berisikan semua makanan / minuman yang dipesan. Juga ada method handlePayment(double amount, ArrayList<Pesanan> pesananList) yang berfungsi untuk menangani proses pembayarn. Di dalam restoran kami, kita bisa membayar secara cash maupun debit card untuk customer agar customer dapat membayar makanan yang sudah mereka pesan.

```
// Terdapat method untuk menjumlahkan total harga dari semua pesanan item
public static double hitungTotal(ArrayList<Pesanan> pesananList) {
    double total = 0;
    for (Pesanan pesanan : pesananList) {
        total += pesanan.getHarga() * pesanan.getJumlah();
    }
    return total;
}
```

```
// Terdapat method untuk memprint struk pesanan
public static void cetakStruk(ArrayList<Pesanan> pesananList) {
    double totalHarga = 0;
    System.out.println(x:"=====");
    System.out.println(x: " Struk Pesanan ");
    System.out.println(x:"=====");
    System.out.printf(format:"%-20s %-10s %-10s%n", ...args:"Nama Menu", "Harga", "Jumlah");
    System.out.println(x:"-----");

    // Terdapat loop pada setiap item pesanan dan memprint detail
    for (Pesanan pesanan : pesananList) {
        System.out.printf(format:"%-20s %-10.2f %-10d%n", pesanan.getNama(), pesanan.getHarga(), pesanan.getJumlah());
        totalHarga += pesanan.getHarga() * pesanan.getJumlah();
    }

    System.out.println(x:"-----");
    System.out.printf(format:"Total Harga: %.2f%n", totalHarga);
    System.out.println(x:"=====");
}

// Memanggil method handlePayment untuk memproses pembayaran
handlePayment(totalHarga, pesananList);
}
```

```

// Terdapat objek Transaction dengan method untuk menangani proses pembayaran
public static void handlePayment(double amount, ArrayList<Pesanan> pesananList) {
    Scanner scanner = new Scanner(System.in);
    boolean paymentSuccess = false;

    // Loop hingga pembayaran berhasil diproses
    while (!paymentSuccess) {
        System.out.println("=====");
        System.out.println("Choose Payment Method");
        System.out.println("=====");
        System.out.println("1. Cash");
        System.out.println("2. Debit");
        System.out.println("-----");
        System.out.print("Enter your choice: ");
        int method = scanner.nextInt();

        // Menangani pembayaran Cash
        if (method == 1) {
            System.out.println("=====");
            System.out.println(amount + " has been paid using cash.");
            System.out.println("=====");
            paymentSuccess = true;
        // Menangani pembayaran kartu debit
        } else if (method == 2) {
            System.out.println("=====");
            System.out.print("Enter your debit card number (12 digits): ");
            String cardNumber = scanner.next();
            if (cardNumber.length() == 12 && cardNumber.matches(regex:"\\d+")) {
                System.out.println(amount + " has been paid using debit card.");
                System.out.println("=====");
                paymentSuccess = true;
            // Jika nomor kartu tidak sesuai
            } else {
                System.out.println("\n-----");
                System.out.println("Payment failed. Invalid card number.");
                System.out.println("-----\n");
            }
            // Jika method pembayaran tidak sesuai
        } else {
            System.out.println("-----");
            System.out.println("Invalid payment method selected.");
            System.out.println("-----");
        }
    }

    // Menghapus daftar pesanan setelah pembayaran berhasil
    if (paymentSuccess) {
        pesananList.clear();
    }
}

```

Proses Transaksi :

Transaksi dalam sistem restoran dilakukan secara sistematis agar mencegah adanya kesalahan di dalam pemesanan makanan ataupun minuman.

- Pelanggan tiba dan melihat menu - menu yang diberikan.
- Pelanggan melakukan pemesanan (objek Pesanan dibuat, yang berisi item dari Menu berisikan nama makanan / minuman, harga dan jumlah).
- Pelanggan memilih meja makan untuk tempat mereka makan, pelanggan juga bisa mengcancel meja yang mereka ingin duduki. Setelah itu pelanggan mengambil nomor antrian yang ada dan mengkonfirmasi apakah pelanggan tersebut VIP atau tidak.
- Pesanan dikirim ke dapur (Koki memproses pesanan dengan interface Iprepare).
- Pesanan disiapkan oleh koki menggunakan interface Iservefood dan dikirim ke pelanggan.
- Pelanggan membayar pesanan yang sudah mereka pesan (cash / debit).

```
Welcome to Five Spices in PIK
.....
1. Display Restaurant Info
2. Display Food Menu
3. Order Food
4. Display Order
5. Display Table Info
6. Print Receipt
7. View Chef Shift Work
8. View Activity Log
9. Keluar
Pilih menu (1-9): 2
Silahkan Pilih Tipe Menu :
1. Appetizer
2. Main Course
3. Dessert
4. Drinks
Masukkan Nomor (1-4):
1
=====
Current Menu
=====
Foods Name    : Caesar Salad
Foods Price   : Rp.20.000
Foods Rating  : 4.2
Foods Type    : Appetizer
Fav's Menu     : Yes
-----
Press 'n' for next menu, 'p' for previous menu, or 'q'
|
```

```
Welcome to Five Spices in PIK
.....
1. Display Restaurant Info
2. Display Food Menu
3. Order Food
4. Display Order
5. Display Table Info
6. Print Receipt
7. View Chef Shift Work
8. View Activity Log
9. Keluar
Pilih menu (1-9): 3
Masukkan tipe Makanan yang ingin Dipesan :
1. Appetizer
2. Main Course
3. Dessert
4. Drinks
Masukkan Nomor (1-4):1
-----
Masukkan Nama Makanan : Caesar Salad
Ilham have started to prepared the dish Caesar Salad

Masukkan Harga Makanan : 20000
Masukkan Jumlah yang ingin dipesan: 2
Ilham are done with Caesar Salad please wait until your Queue is up.
-----
```

```
Pilih menu (1-9): 5
Table Information:
-----
Nomor Meja      Status
Meja 1          Available
Meja 2          Available
Meja 3          Available
Meja 4          Available
Meja 5          Available
Meja 6          Available
Meja 7          Available
Meja 8          Occupied
Meja 9          Occupied
Meja 10         Occupied
-----
Pilih nomor meja untuk makan: 5
Meja Meja 5 telah berhasil dipesan.
Apakah anda ingin merubah meja anda?
n
Ambil Nomor Antrian? :
1. Ya
2. Tidak
1
Masukan Nama Anda : Darrell
Apakah Kamu Member VIP? (y/n) :
n
```

```
Pilih Aksi Anda :
1. Tampilkan Antrian
2. Dequeue Antrian
3. Tidak (Quit)
1

| Nama      | NoAntri | VIP? |
?????????????????????????????????
| Kobe      | 1       | No   |
| Jampar    | 2       | No   |
| Lisa      | 3       | No   |
| Darrell   | 4       | No   |
Pilih Aksi Anda :
1. Tampilkan Antrian
2. Dequeue Antrian
3. Tidak (Quit)
3
```

```
Welcome to Five Spices in PIK
-----
1. Display Restaurant Info
2. Display Food Menu
3. Order Food
4. Display Order
5. Display Table Info
6. Print Receipt
7. View Chef Shift Work
8. View Activity Log
9. Keluar
Pilih menu (1-9): 6
=====
Struk Pesanan
=====
Nama Menu      Harga     Jumlah
-----
Caesar Salad   20000.00  2
-----
Total Harga: 40000.00
=====
Choose Payment Method
=====
1. Cash
2. Debit
-----
Enter your choice: 1
-----
40000.0 has been paid using cash.
=====
```

3. Question 3: Sub-CLO 3, Weight(56.24%)

In the Final Project Assignment, make sure the code contain these following points:

- a. CRUD (Create, Read, Delete, Update) for the master objects
- b. CRUD for transaction objects involving at least 2 other objects
- c. Exception handling to prevent users from inputting values that do not match the format, or catching the errors
- d. Data is stored in a linked list data structure (single, double, or circular)
- e. There is one use of Stack or Queue in the project

Answer :

a&b

Create : Dalam proses transaksi kami, makanan yang sudah dipesan akan di create struk atau billing dari makanan yang sudah dipesan. Struk yang buat akan berisikan nama makanan, jumlah makanan dan harga makanan serta total makanan/minuman yang harus dibayar.

```
// Terdapat method untuk memprint struk pesanan
public static void cetakStruk(ArrayList<Pesanan> pesananList) {
    double totalHarga = 0;
    System.out.println(x:"=====");
    System.out.println(x: "           struk Pesanan          ");
    System.out.println(x:"=====");
    System.out.printf(format:"%-20s %-10s %-10s%n", ...args:"Nama Menu", "Harga", "Jumlah");
    System.out.println(x:"-----");

    // Terdapat loop pada setiap item pesanan dan memprint detail
    for (Pesanan pesanan : pesananList) {
        System.out.printf(format:"%-20s %-10.2f %-10d%n", pesanan.getNama(), pesanan.getHarga(), pesanan.getJumlah());
        totalHarga += pesanan.getHarga() * pesanan.getJumlah();
    }

    System.out.println(x:"-----");
    System.out.printf(format:"Total Harga: %.2f%", totalHarga);
    System.out.println(x:"=====");

    // Memanggil method handlePayment untuk memproses pembayaran
    handlePayment(totalHarga, pesananList);
}
```

Read : Dalam proses transaksi kami, kita sebagai user dapat menampilkan makanan apa saja yang sudah kita pesan. method ini dapat dilakukan di kelas Main dengan method 4. Display Order. Disini code akan membaca dan menampilkan makanan apa saja yang sudah dipesan.

```

while (true) {
    System.out.println("-----");
    System.out.println("Display Order");
    System.out.println("-----");
    // Hanya menampilkan daftar pesanan tanpa mencetak struk
    for (Pesanan pesanan : pesananList) {
        System.out.println("-----");
        System.out.println("Nama Menu: " + pesanan.getNama() + ", Harga: " + pesanan.getHarga() + ", Jumlah: " + pesanan.getJumlah());
    }
    System.out.println("-----");
    System.out.println("1. Add More Order");
    System.out.println("2. Remove Order");
    System.out.println("3. Quit");
    System.out.println("-----");
    System.out.print("Pilihan: ");
}

```

Update : Dalam proses transaksi kami, kita sebagai user dapat mengubah / mengupdate makanan yang sudah dipesan. Semisal, pelanggan (user) telah memesan makanan yang bernama Caesar Salad. Namun tidak jadi membeli dan ingin memakan Steak. Maka user dapat mengubah order makanannya sesuai selera nya.

```

// Terdapat method untuk menangani pesanan makanan
public void handleOrderFood(Scanner scan, ArrayList<Pesanan> pesananList) {
    // Meminta pelanggan memasukkan nama makanan
    System.out.print("Masukkan Nama Makanan : ");
    String namaMakanan = scan.nextLine();

    if (checkStock(namaMakanan)) {
        // Meminta pelanggan memasukkan harga makanan
        System.out.print("Masukkan Harga Makanan : ");
        double hargaMakanan = scan.nextDouble();

        // Meminta pelanggan untuk memasukkan jumlah makanan yang ingin dipesan
        System.out.print("Masukkan Jumlah yang ingin dipesan: ");
        int jumlahMakanan = scan.nextInt();
        scan.nextLine();

        // Menambahkan pesanan baru ke list pesanan
        pesananList.add(new Pesanan(namaMakanan, hargaMakanan, jumlahMakanan));

        // Mengurangi stok makanan yang dipesan
        decreaseStock(namaMakanan, jumlahMakanan);

        // Akan keluar output bahwa pesanan berhasil ditambahkan
        System.out.println("Pesanan berhasil ditambahkan.");
    } else {
        // Akan keluar output bahwa makanan tidak tersedia
        System.out.println("Maaf, makanan tidak tersedia.");
    }
}

```

```
Welcome to Five Spices in PIK
.....
1. Display Restaurant Info
2. Display Food Menu
3. Order Food
4. Display Order
5. Display Table Info
6. Print Receipt
7. View Chef Shift Work
8. View Activity Log
9. Keluar
Pilih menu (1-9): 4
=====
Display Order
=====
-
Nama Menu: Caesar Salad, Harga: 20000.0, Jumlah: 3
-
1. Add More Order
2. Remove Order
3. Quit
-----
```

```
Pilihan: 1
Masukkan Nama Makanan : Steak
Masukkan Harga Makanan : 50000
Masukkan Jumlah yang ingin dipesan: 2
Stok Steak berkurang 2.

Pesanan berhasil ditambahkan.
=====
Display Order
=====
-
Nama Menu: Caesar Salad, Harga: 20000.0, Jumlah: 3
-
Nama Menu: Steak, Harga: 50000.0, Jumlah: 2
-
1. Add More Order
2. Remove Order
3. Quit
-----
Pilihan: 2
Masukkan Nama Makanan yang ingin dihapus: Caesar Salad
Pesanan telah dihapus.
=====
Display Order
=====
-
Nama Menu: Steak, Harga: 50000.0, Jumlah: 2
-
1. Add More Order
2. Remove Order
3. Quit
-----
```

Delete : Dalam proses transaksi kami, kita sebagai user dapat mendelete makanan yang tidak mau / tidak jadi kita pesan.

```

System.out.println(x:-----");
System.out.println(x:"1. Add More Order");
System.out.println(x:"2. Remove Order");
System.out.println(x:"3. Quit");
System.out.println(x:-----");
System.out.print(s:"Pilihan: ");
int submenuChoice = scan.nextInt();
scan.nextLine(); // consume newline
switch (submenuChoice) {
    case 1:
        menuStock.handleOrderFood(scan, pesananList);
        break;
    case 2:
        System.out.print(s:"Masukkan Nama Makanan yang ingin dihapus: ");
        String namaMakanan = scan.nextLine();
        boolean found = false;
        for (Pesanan pesanan : pesananList) {
            if (pesanan.getNama().equalsIgnoreCase(namaMakanan)) {
                pesananList.remove(pesanan);
                found = true;
                System.out.println(x:"Pesanan telah dihapus.");
                break;
            }
        }
}

```

c. Implementation of Exception Handling

Exception Handling digunakan untuk mengelola potensi kesalahan yang dapat terjadi selama input pengguna. Fokus utama di sini adalah menangkap InputMismatchException yang merupakan masalah umum saat berurusan dengan input pengguna dalam aplikasi konsol. Berikut adalah implementasi Exception Handling di code kami

```

98     try {
99         int choice = scan.nextInt();
100        switch (choice) {
101
} catch (InputMismatchException e) {
    System.out.println(x:"Input tidak valid. Silakan masukkan data tipe angka.");
    scan.next();
}

```

Block Try-Catch:

Block Try: Block ini berisi kode yang berpotensi untuk melemparkan sebuah exception. Dalam kasus ini, kode yang berpotensi melemparkan exception adalah `scan.nextInt()` dan metode input lainnya di mana pengguna diharapkan untuk memasukkan data.

Blok Catch: Blok ini menangkap exception spesifik (dalam hal ini, `InputMismatchException`) dan menanganinya dengan cara yang sesuai.

Handling `InputMismatchException`:

`InputMismatchException`: Exception ini dilemparkan oleh metode kelas `Scanner` ketika input tidak sesuai dengan tipe data yang diharapkan. Misalnya, jika program mengharapkan sebuah bilangan bulat (`integer`) dan pengguna memasukkan sebuah string, maka exception ini akan dilemparkan, dan akan mengeluarkan output “Input tidak valid. Silahkan masukkan tipe data angka”

d. Implementation of Linked List

Single Linked List merupakan struktur data fundamental yang digunakan dalam Computer Science untuk mengorganisasi elemen yang biasa disebut node terhubung satu sama lain menggunakan referensi ataupun link pada linear. Setiap node memiliki dua bagian utama yaitu data terdiri atas nilai dan pointer menunjuk ke node berikutnya dalam linked list. Terdiri atas dua bagian, bagian depan disebut head dan bagian belakang disebut tail.

Kegunaan Linked List yaitu ukurannya dinamis yang mudah kurang dan tambah tanpa alokasi memori besar dari awal. Kemudian ada insert dan delete data yang efisien karena elemen ini hanya perlu disambungkan ulang dan tidak ada batasan ukuran yang tetap seperti array. Lalu mengimplementasikan struktur data meliputi stack, queue. Terdapat juga kelemahan karena akses acak tidak efisien dan overhead penyimpanan tambahan untuk setiap tautan sehingga didasarkan kebutuhan spesifik dari masalah yang dihadapi. Operasinya terdiri atas traversal untuk mengetahui jumlah data yang kita punya di linked list, insertion menambah sebuah node depan, tengah dan belakang serta stack mencari data sesuai kriteria.

```
// Terdapat method untuk menampilkan informasi dari koki
public void displayKokiInfo (){
    System.out.println("Name : " + getName());
    System.out.println("Position : " + getPosition());
    System.out.println("Speciality : " + getSpeciality());
    System.out.println("Experience : " + getExperience());
    System.out.println("Certified Chef : " + isCertified());
    System.out.println(x:"");
}
```

```
private void addToLinkedList(Koki koki) {
    // Jika head masih null, maka koki baru ini menjadi head dari linked list
    if (head == null) {
        head = koki;
        // Jika head sudah ada, maka kita mulai dari head
    } else {
        Koki current = head;
        // Iterasi hingga menemukan node terakhir (next == null)
        while (current.next != null) {
            current = current.next;
        }
        // Menambahkan koki baru di akhir linked list
        current.next = koki;
    }
}
```

```
// SINGLE LINKED LIST MEMAKAI NEXT.
public static void displayAllKokiInfo() {
    System.out.println(x:-----");
    System.out.println(x:      Our Chefs      ");
    System.out.println(x:-----");
    Koki current = head;
    while (current != null) {
        current.displayKokiInfo();
        current = current.next;
    }
}
```

Dalam Code kami, kami memakai Singly Linked List ditunjukkan lewat method “addToLinkedList(Koki koki) dengan menambahkan data ke node baru ‘koki’ ke akhir single linked list dan “displayAllKokiInfo()” untuk melakukan iterasi dan menampilkan informasi dari setiap node dalam single linked list. Dimana dalam konteks ini ditunjukkan head sebagai node pertama dan mereka memiliki referensi ke node berikutnya serta proses iterasi dilakukan hingga mencapai node terakhir.

Sumber: [Singly Linked List Tutorial - GeeksforGeeks](#)

Doubly Linked List merupakan pengembangan dari Single Linked List dimana linknya ada dua bagian yaitu next dan previous. Jadinya head berarti previous null dan tail menuju null merupakan data terakhir. Terdapat juga push untuk insert front dan append untuk insert back. Kegunaan Doubly Linked List yaitu pengimplementasiannya digunakan sebagai dasar struktur data lain seperti stack, queue dan dequeue. Lalu penambahan ataupun penghapusan elemen yang mudah tanpa memerlukan alokasi memori besar dari awal serta tidak ada batasan ukuran tetap seperti array.

```
// Referensi menuju ke node pertama dalam daftar terhubung (log paling baru)
private LogNode head;

// Referensi menuju ke node terakhir dalam daftar terhubung (log paling lama)
private LogNode tail;

// Referensi menuju ke node yang sedang dipilih saat ini
private LogNode current;

public void addLog(int day, int customers, double revenue) {
    LogNode newNode = new LogNode(day, customers, revenue);
    if (head == null) {
        head = newNode;
        tail = newNode;
    } else {
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }
}

public void printCurrentLog() {
    if (current != null) {
        System.out.println("Log" + (current.day == 0 ? " Hari ini : " - " + current.day + " hari yang lalu"
        System.out.println("Jumlah pelanggan : " + current.customers);
        System.out.println("Pendapatan: Rp " + current.revenue);
        System.out.println(x:"=====");
    } else {
        System.out.println(x:"Log kosong.");
    }
}
```

Dalam Code kami, kami memakai Doubly Linked List di dalam class ActivityLog ditunjukkan oleh deklarasi kelas Node dimana menggunakan referensi ‘next’ ditujukan pada node selanjutnya dan ‘prev’ untuk node sebelumnya. Kelas AvtivityLog merupakan class yang berfungsi untuk menyimpan data Revenue Restoran untuk 7 hari kebelakang (History). Menjadikan Owner restoran dapat melihat revenue restoran untuk 7 hari ke belakang. Kemudian mendeklarasikan atribut ‘LogNode’ yang sama halnya seperti tadi terdiri dari ‘head’, ‘tail’ serta ‘current’ untuk melacak node pertama, terakhir dan node saat ini. Terdapat method ‘addLog’ yang menambahkan node baru di awal list dan mengatur referensi ‘next’ dan ‘prev’ untuk menjaga hubungan dua arah dan juga method ‘printCurrentLog’ untuk menampilkan informasi dari node yang saat ini dipilih.

Sumber: [Introduction to Doubly Linked List – Data Structure and Algorithm Tutorials - GeeksforGeeks](#)

Single Circular Linked List merupakan struktur data linear terdiri dari sekelompok simpul yang terhubung secara siklik maksudnya simpul terakhir yang ada pada linked list terhubung kembali ke simpul pertama membentuk sebuah lingkaran. Jadi, setiap simpul memiliki dua bagian yaitu data dan tautan ke simpul berikutnya dalam urutan. Kegunaan Circular Linked List yaitu mengelola data dalam bentuk lingkaran, mengalokasikan memori dengan cara efisien karena dapat mengelola daftar blok memori yang tersedia dan terpakai serta mengimplementasikan struktur data terhubung berulang.

```
// Terdapat method untuk menampilkan nama, posisi, keahlian, dan waktu shift dari setiap Koki
public void viewShiftKoki() {
    System.out.println(x:"-----");
    System.out.println(x:"      Koki Shifts      ");
    System.out.println(x:"-----");
    // Menampilkan informasi tentang koki
    System.out.println("Name: " + getName()); // Menampilkan nama
    System.out.println("Position: " + getPosition()); // Menampilkan posisi
    System.out.println("Speciality: " + getSpeciality()); // Menampilkan keahlian
    System.out.println("Shift Time: " + getShiftWaktu()); // Menampilkan waktu shift
    System.out.println(x:"_");
}
```

```
public class ShiftKoki extends Koki {
    // Terdiri atas atribut privat untuk menyimpan waktu shift
    private String shiftWaktu;

    // Terdapat konstruktor untuk menginisialisasi objek ShiftKoki
    public ShiftKoki(String name, String position, String speciality, int experience, boolean isCertified,
                      MenuStock menuStock, String shiftWaktu) {
        // Memanggil constructor superclass (Koki)
        super(name, position, speciality, experience, isCertified, menuStock);
        // Menginisialisasi atribut shiftWaktu
        this.shiftWaktu = shiftWaktu;
    }
}
```

```
public class ShiftSCLList {
    // Terdiri atas node privat saat ini
    private Node current;

    // Terdapat kelas privat Node untuk single circular linked list
    private class Node {
        // Terdiri atas data berupa objek ShiftKoki
        ShiftKoki data;
        // Terdiri referensi ke node selanjutnya
        Node next;

        // Terdapat konstruktor untuk Node
        public Node(ShiftKoki data) {
            this.data = data;
            this.next = null;
        }
    }
}
```

```

// Terdapat method untuk menambahkan shift ke dalam list
public void addShift(ShiftKoki shift) {
    Node newNode = new Node(shift);
    if (isEmpty()) {
        head = newNode;
        newNode.next = head;
        // Mengatur current ke head yang baru ditambahkan
        current = head;
    } else {
        Node temp = head;
        while (temp.next != head) {
            temp = temp.next;
        }
        temp.next = newNode;
        newNode.next = head;
    }
    size++;
}

// Terdapat method untuk menampilkan informasi shift saat ini
public void displayCurrentShift() {
    // Menampilkan informasi shift koki
    if (current != null) {
        current.data.viewShiftKoki();
    } else {
        System.out.println("Shift not available.");
    }
}

```

Dalam Code kami, kami memakai Class turunan dari Koki yang bernama ShiftKoki. Kelas ShiftKoki memiliki atribut baru yaitu String shiftWaktu. Kami membuat method addShift untuk memasukan data shift koki yang akan dimasukan ke Single Circular Linked List. Kami menggunakan Single Circular Linked List yang berfungsi untuk mencetak informasi Shift koki yang ada. Dengan menggunakan fitur dari Single Circular Linked List, customer sebagai user bisa melihat shift yang ada dan menekan next untuk melihat shift selanjutnya.

Sumber: [Circular Linked List Implementation in Java - GeeksforGeeks](#)

Circular Double Linked List merupakan struktur data linear terdiri dari sekelompok simpul membentuk lingkaran dan tiap simpul memiliki navigasi dua link yaitu previous dan next untuk ke simpul sebelumnya dan berikutnya disesuaikan dengan arah panahnya. Kegunaan Circular Doubly Linked List yaitu memungkinkan untuk traversing maju dan mundur secara efisien maksudnya menelusuri node sampai selesai dan setiap link harus terhubung serta meningkatkan efisiensi dalam

penggunaan cache karena data yang sering diakses memungkinkan akses cepat ke data baik dari depan maupun belakang.

```
private static class MenuItem {
    Menu menu;
    MenuItem next;
    MenuItem prev;

    MenuItem(Menu menu) {
        this.menu = menu;
        this.next = null;
        this.prev = null;
    }
}
```

```
// Mendeklarasikan kelas MenuCDLList
public class MenuCDLList {
    private MenuItem head;
    private MenuItem current;

    // Terdapat method untuk menambahkan menu ke dalam linked list
    public void addMenu(Menu menuItem) {
        MenuItem newNode = new MenuItem(menuItem);
        if (head == null) {
            head = newNode;
            head.next = head;
            head.prev = head;
            current = head;
        } else {
            MenuItem last = head.prev;
            last.next = newNode;
            newNode.prev = last;
            newNode.next = head;
            head.prev = newNode;
        }
    }
}
```

```
// Terdapat method untuk menampilkan menu saat ini
public void displayCurrentMenu() {
    if (current != null) {
        System.out.println("=====");
        System.out.println("          Current Menu");
        System.out.println("=====");
        current.menu.displayMenu();
        System.out.println("-----");
    } else {
        System.out.println("Menu not available.");
    }
}
```

```
// Menambahkan menu ke daftar menu
// Appetizer
menuListAp.addMenu(appettizer1);
menuListAp.addMenu(appettizer2);
menuListAp.addMenu(appettizer3);
menuListAp.addMenu(appettizer4);
menuListAp.addMenu(appettizer5);
```

Dalam Code kami, kami menggunakan Circular Double Linked List ditunjukkan oleh deklarasi kelas MenuCDLL yang mendefenisikan node dari kelas MenuItem yang menginisialisasi atribut ‘menu’ yang menyimpan objek Menu mewakili item menu, ‘next’ dan ‘prev’ diatur ke referensi this menandakan node pertama dan terakhir dalam Linked List terhubung satu sama lain. Menjadikan Linked list dapat menunjuk ke node next, prev dan mengulang ke node pertamanya kembali. Di dalam code kami, kami menggunakan Circular Double Linked List untuk mencetak menu menu yang ada di dalam code kami. Menu yang nanti dipilih dengan user juga bervariasi tergantung dengan tipe menu, yaitu Appetizer, MainCourse, Dessert, Drinks. User nanti dapat mencetak menu yang dipilih dan bisa melihat menu selanjutnya dengan menekan “n” dan sebelumnya dengan menekan “p”.

Sumber: [Circular Doubly Linked List - javatpoint](#)

e. Implementation Stack or Queue

Stack ataupun tumpukan merupakan bagian dari linked list dan merupakan struktur data yang mengikuti prinsip LIFO (Last In, First Out) maksudnya elemen terakhir dimasukkan ke dalam stack menjadi elemen pertama yang diambil dari stack. Pengoperasiannya sendiri terdiri dari Push untuk menambah elemen kedalam Stack, Pop untuk menyingkirkan elemen teratas dari Stack dan menjadikan yang paling bawah sebagai head. Dilanjutkan dengan Peek / Top dengan melihat elemen paling atas serta isEmpty mengecek stack kosong atau tidak.

Kegunaan Stack yaitu mengelola data dan fungsi secara efisien sama halnya dengan pengontrolan memori dialokasikan atau tidak. Ia memungkinkan dalam penambahan dan pengurangan elemen dengan urutan yang telah ditentukan lewat pengekspresian

postfix atau infix dan digunakan pada berbagai aplikasi untuk manajemen data, fungsi dan memori yang efisien.

```
// Terdapat konstruktor untuk membuat objek Restaurant dengan parameter yang diberikan
public Restaurant(String name, String location, int year, String owner, double rating) {
    // Menginisiasi atribut-atribut dari objek Restaurant sesuai dengan parameter yang diberikan
    this.name = name;
    this.location = location;
    this.year = year;
    this.owner = owner;
    this.rating = rating;
    this.status = "Available"; // Set status awal restoran sebagai "Available"
    this.feedbackStack = new Feedback(); // Inisialisasi feedbackStack
}
```

```
// Mendeklarasikan kelas Feedback
class Feedback {
    // Terdiri atas kelas privat static Node
    private static class Node {
        // Terdiri atas atribut untuk menyimpan feedback dan referensi ke node berikutnya
        String feedback;
        Node next;

        // Terdapat konstruktor Node yang menerima feedback
        Node(String feedback) {
            this.feedback = feedback;
        }
    }

    // Terdapat atribut privat untuk menunjuk ke node teratas dari stack
    private Node top;

    // Terdapat konstruktor Feedback menginisialisasi top sebagai null
    public Feedback() {
        top = null;
    }
}
```

```
// Terdapat method untuk menampilkan semua feedback di stack
public void displayFeedbacks() {
    Node current = top;
    // Jika tidak ada feedback, tampilkan pesan bahwa belum ada feedback
    if (current == null) {
        System.out.println("Belum ada feedback.");
        return;
    }
    // Jika ada feedback, tampilkan semua
    System.out.println("Feedback :");
    while (current != null) {
        System.out.println("-----");
        System.out.println("++ " + current.feedback);
        System.out.println("-----\n");
        current = current.next;
    }
}
```

```
// Meminta pelanggan memasukkan feedback ke restoran
switch (infoChoice) {
    case 1:
        System.out.print(s:"Masukkan feedback Anda: ");
        String feedback = scan.nextLine();
        restaurant.addFeedback(feedback);
        break;
    case 2:
        boolean feedbackMenuQuit = false;
        while (!feedbackMenuQuit) {
            restaurant.displayFeedbacks();
            System.out.println(x:"1. Quit");
            System.out.print(s:"Pilih aksi (1): ");
            int feedbackChoice = scan.nextInt();
            if (feedbackChoice == 1) {
                feedbackMenuQuit = true;
            } else {
                System.out.println(x:"Pilihan tidak valid. Silakan coba lagi.");
            }
        }
        break;
}
```

Dalam Code kami, kami menggunakan Stack ditunjukkan dari deklarasi kelas Feedback yang berada di dalam Class Restaurant. Kelas Feedback mendefenisikan node dalam stack dan setiap node memiliki dua atribut yaitu ‘String feedback’ yang menyimpan feedback dari pelanggan dan ‘next’ merupakan referensi ke node berikutnya dalam stack dan konstruktor Feedback Class Feedback yang berada di dalam Restaurant (Inner Class). Terdapat juga method ‘displayFeedbacks()’ yang mengiterasi stack dan mencatat setiap feedback. Dengan adanya stack, feedback restoran yang dimasukan pertama akan berada menjadi feedback yang paling bawah.

Sumber: [Java Stack - Javatpoint](#)

Queue adalah struktur data yang digunakan untuk menyimpan berbagai elemen yang disusun sesuai prinsip, First In, First Out atau disingkat dari FIFO. Artinya, elemen yang masuk pertama kali pada antrian juga dikeluarkan pertama kali dari antrian. Queue mengikuti prinsip FIFO, yang berarti elemen pertama yang dimasukkan ke dalam queue akan menjadi elemen pertama yang keluar. Hal ini sering kali sangat penting dalam pemrosesan data berurutan, seperti antrian pesan, penjadwalan tugas, atau proses yang membutuhkan penanganan dalam urutan kedatangan.

Dalam OOP, queue dapat diimplementasikan dengan mudah menggunakan struktur data sederhana seperti array atau linked list. Hal ini membuatnya mudah dipahami dan diterapkan dalam berbagai konteks pemrograman.

```
class Customer {
    String nameCust;
    int nomorAntrian;
    boolean isVip;

    Customer(String nameCust, int nomorAntrian, boolean isVip) {
        this.nameCust = nameCust;
        this.nomorAntrian = nomorAntrian;
        this.isVip = isVip;
    }

    @Override
    public String toString() {
        return String.format("%-10s | %-3d | %-3s |", nameCust, nomorAntrian, isVip ? "Yes" : "No");
    }
}
```

```
public class FiveSpicesQueue {
    private Node front;
    private Node rear;
    /*
    Konstruktor FiveSpicesQueue berperan dalam
    objek antrian dibuat dengan menginisialisasi
    pada awalnya, antrian tersebut kosong */
    public FiveSpicesQueue() {
        this.front = null;
        this.rear = null;
    }
```

```
public void displayQueue() {
    Node temp = front;
    while (temp != null) {
        System.out.println(temp.customer);
        temp = temp.next;
    }
}
```

Dalam code, kami menggunakan Queue dimana ditunjukkan oleh deklarasi dari kelas Customer yang memiliki atribut String nameCust, int nomorAntrian, boolean isVip dan juga kelas FiveSpicesQueue yang mengatur antrian berisi Customer yang memiliki dua atribut privat yaitu ada ‘front’ dan ‘rear’ merujuk pada node pertama dan terakhir dalam antrian. Konstruktor FiveSpicesQueue menginisialisasi kedua atribut tersebut null bahwa antrian kosong. Kemudian terdapat method Enqueue yang menambahkan customer baru ke antrian diikuti method Dequeue yang menghapus customer pertama dari antrian. Terdapat kedua method yaitu ada isEmpty untuk memeriksa apakah antrian kosong atau tidak dan displayQueue yang mencetak isi antrian. Sumber: [Queue Data Structure - GeeksforGeeks](#)

UML DIAGRAM

