LAPORAN

RESPONSI PBO

RAFLI HAIKAL PUTRA

5230411301

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2024

Soal Teori

1.Jelaskan perbedaan use case diagram dengan class diagram?

2.Jelaskan jenis-jenis dependensi?

3.Apa perbedaan pemrograman terstruktur dengan berorientasi objek, jelaskan?

4.Jelaskan konsep objek dan beri contohnya?

5.Jelaskan jenis-jenis access modifier beri contohnya dalam baris pemrograman?

6.Gambarkan contoh pewarisan dalam diagram class?

JAWABAN:

1.Perbedaan use case diagram dengan class diagram:

•Use case diagram menggambarkan interaksi antara aktor (pengguna atau sistem lain) dengan sistem yang dibuat. Diagram ini menunjukkan fungsionalitas yang disediakan sistem sesuai dengan kebutuhan pengguna.

•Class diagram menggambarkan struktur sistem dalam bentuk kelas dan hubungan antar kelas tersebut. Diagram ini menunjukkan atribut, metode, dan relasi antar kelas seperti pewarisan, asosiasi, dan komposisi.

2.Jenis-jenis dependensi:

•Association: Hubungan antara dua kelas yang menunjukkan bahwa satu kelas menggunakan atau mengetahui kelas lainnya.

•Aggregation: Hubungan di mana satu kelas berfungsi sebagai bagian dari kelas lain namun masih dapat berdiri sendiri.

•Composition: Hubungan di mana satu kelas berfungsi sebagai bagian dari kelas lain dan tidak dapat berdiri sendiri jika kelas utama dihapus.

•Inheritance: Hubungan di mana satu kelas mewarisi atribut dan metode dari kelas induk.

3.Perbedaan pemrograman terstruktur dengan berorientasi objek:

•Pemrograman terstruktur berfokus pada prosedur atau fungsi untuk mengorganisir kode. Pendekatan ini memecah program menjadi fungsi-fungsi kecil yang bisa digunakan ulang.

•Pemrograman berorientasi objek berfokus pada objek yang menggabungkan data dan metode yang beroperasi pada data tersebut. Pendekatan ini menggunakan konsep seperti kelas, objek, pewarisan, polimorfisme, enkapsulasi, dan abstraksi.

4.Konsep objek dan contohnya:

•Objek adalah entitas nyata atau abstrak yang memiliki atribut (data) dan metode (fungsi) dalam konteks pemrograman berorientasi objek. Objek merupakan instansiasi dari kelas.

•Contoh: Dalam kelas “Mobil”, sebuah objek bisa berupa mobil tertentu seperti “MobilA” dengan atribut seperti warna, merk, dan kecepatan, serta metode seperti “jalan()” atau “rem()”.

5.Jenis-jenis access modifier dan contohnya dalam baris pemrograman:

•Public: Dapat diakses dari mana saja.

•Protected: Dapat diakses oleh kelas itu sendiri dan subclass.

•Private: Hanya dapat diakses oleh kelas itu sendiri.

•Contoh:

```
public class Mobil {
   private String merk;
   protected int kecepatan;
   public void jalan() {
      // logika jalan
   }
}
```

6. Contoh pewarisan dalam diagram class:

Untuk menggambar pewarisan, buatlah diagram yang menunjukkan satu kelas sebagai superclass dan kelas lainnya sebagai subclass yang memiliki tanda panah ke atas menuju superclass. Misalnya:

Kendaraan <---- Mobil

Kendaraan <---- Motor

Soal Praktik

Implementasikan class diagram di atas ke dalam Python dengan ketentuan sebagai berikut:

1. Gunakan Overriding dalam Class Snack, Makanan, dan Minuman.
2. Method yang digunakan silakan tambahkan masing-masing.

Jawaban Implementasi dalam Python

Berikut adalah kode Python yang mengimplementasikan diagram kelas tersebut, dengan menerapkan method overriding di kelas Snack, Makanan, dan Minuman.

```
class Pegawai:
    def __init__(self, nik, nama, alamat):  # Memperbaiki __init__ dari _init_
        self.nik = nik
        self.nama = nama
        self.alamat = alamat

class Transaksi:
    def __init__(self, no_transaksi, detail_transaksi):  # Memperbaiki __init__ dari _init_
        self.no_transaksi = no_transaksi
        self.detail_transaksi = detail_transaksi

class Struk:
    def __init__(self, no_transaksi, nama_pegawai, no_transaksi_detail, nama_produk, jumlah_produk, total_harga):  # Memperbaiki __init__ dari _init_
        self.no_transaksi = no_transaksi
        self.nama_pegawai = nama_pegawai
        self.no_transaksi_detail = no_transaksi_detail
        self.nama_produk = nama_produk
```

```
        self.jumlah_produk = jumlah_produk
        self.total_harga = total_harga

    def print_struk(self):
        print("=== Struk Pembelian ===")
        print(f"No Transaksi: {self.no_transaksi}")
        print(f"Nama Pegawai: {self.nama_pegawai}")
        print(f"No Transaksi Detail: {self.no_transaksi_detail}")
        print(f"Nama Produk: {self.nama_produk}")
        print(f"Jumlah Produk: {self.jumlah_produk}")
        print(f"Total Harga: Rp {self.total_harga}")
        print("========================")

class Produk:
    def __init__(self, kode_produk, nama_produk, jenis_produk):  # Memperbaiki __init__ dari _init_
        self.kode_produk = kode_produk
        self.nama_produk = nama_produk
        self.jenis_produk = jenis_produk

    def get_info(self):
        return f"Kode Produk: {self.kode_produk}, Nama Produk: {self.nama_produk}, Jenis Produk: {self.jenis_produk}"

class Snack(Produk):
    def __init__(self, kode_produk, nama_snack, harga):  # Memperbaiki __init__ dari _init_
        super().__init__(kode_produk, nama_snack, "Snack")  # Memperbaiki __init__ dari _init_
        self.harga = harga

    def get_info(self):  # Overriding method
```

```
        return f"Snack: {self.nama_produk}, Harga: Rp {self.harga}"

class Makanan(Produk):
    def __init__(self, kode_produk, nama_makanan, harga):  # Memperbaiki __init__ dari _init_
        super().__init__(kode_produk, nama_makanan, "Makanan")  # Memperbaiki __init__ dari _init_
        self.harga = harga

    def get_info(self):  # Overriding method
        return f"Makanan: {self.nama_produk}, Harga: Rp {self.harga}"

class Minuman(Produk):
    def __init__(self, kode_produk, nama_minuman, harga):  # Memperbaiki __init__ dari _init_
        super().__init__(kode_produk, nama_minuman, "Minuman")  # Memperbaiki __init__ dari _init_
        self.harga = harga

    def get_info(self):  # Overriding method
        return f"Minuman: {self.nama_produk}, Harga: Rp {self.harga}"

# Contoh penggunaan
pegawai = Pegawai("123", "Budi", "Jl. Merdeka")
transaksi = Transaksi("001", "Detail transaksi")
struk = Struk("001", pegawai.nama, transaksi.no_transaksi, "Nasi Goreng", 3, 60000)
snack = Snack("S001", "Keripik", 5000)
makanan = Makanan("M001", "Nasi Goreng", 20000)
minuman = Minuman("D001", "Teh Manis", 7000)

# Output informasi
print(snack.get_info())
print(makanan.get_info())
```

```
print(minuman.get_info())

print()

struk.print_struk()
```

Penjelasan:

1.Overriding: Method get_info() di kelas Produk di-override dalam kelas Snack, Makanan, dan Minuman untuk menampilkan informasi yang spesifik sesuai dengan jenis produk.

2.Penggunaan Method Tambahan: Method print_struk() di Struk digunakan untuk mencetak detail transaksi struk.