RESPONSI

IMPLEMENTASI PROGRAM BERORIENTASI OBJEK

Disusun Untuk Memenuhi Tugas

Mata Kuliah: Pemrograman Berorientasi Objek Praktik

Dosen Pembimbing:
Sri Wulandari, S.Kom., M.Cs.

Disusun oleh:
Muhammad Zaki As Shidiqi

5240411230

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2025

A. Latihan 1

1. Kode Program

```
class Akun:
    def __init__(self, saldo):
        self.saldo = saldo

    def setor(self, jumlah):
        if jumlah <= 0:
            print('Jumlah tidak valid')
        else:
            self.saldo += jumlah

    def tarik(self, jumlah):
        if jumlah > self.saldo:
            print('Jumlah tidak valid')
        else:
            self.saldo -= jumlah


class AkunReward(Akun):
    def __init__(self, saldo):
        super().__init__(saldo)
        self.poin = 0

    def setor(self, jumlah):
        super().setor(jumlah)
        if jumlah <= 1000:
            self.poin += 1
        elif 1001 <= jumlah <= 5000:
            self.poin += 5
        elif jumlah > 5000:
            self.poin += 10
        print(f'Setor: {jumlah} -> saldo = {self.saldo}, poin = {self.poin}')


    def tarik(self, jumlah):
        super().tarik(jumlah)
        if jumlah <= 1000:
            self.poin -= 1
        elif jumlah > 1000:
            self.poin -= 3
        print(f'Tarik: {jumlah} -> saldo = {self.saldo}, poin = {self.poin}')

akun = AkunReward(0)
akun.setor(1000)
akun.tarik(500)
```

2. Output

```
Setor: 1000 → saldo = 1000, poin = 1
Tarik: 500 → saldo = 500, poin = 0
```

3. Penjelasan

Pada latihan 1 ini, saya membuat dua class yaitu “Akun” dan “AkunReward”. Class “Akun” berperan sebagai parent class yang merepresentasikan akun dasar dengan fitur umum seperti menyetor dan menarik saldo. Sedangkan class “AkunReward” merupakan child class yang mewarisi seluruh atribut dan method dari class “Akun”, namun memiliki tambahan fitur sistem poin setiap kali pengguna melakukan transaksi.

Class Akun memiliki satu atribut yaitu saldo, yang digunakan untuk menyimpan jumlah uang yang dimiliki oleh akun. Terdapat dua method utama, yaitu setor() untuk menambah saldo dan tarik() untuk mengurangi saldo. Kedua method ini dilengkapi dengan validasi agar jumlah yang dimasukkan tidak bernilai negatif atau melebihi saldo yang tersedia.

Class AkunReward menambahkan atribut baru yaitu poin, yang digunakan untuk menyimpan jumlah poin reward. Pada method setor(), selain menambah saldo seperti di parent class, sistem juga memberikan poin sesuai jumlah uang yang disetor. Semakin besar jumlah setorannya, semakin banyak poin yang diterima. Method tarik() juga dioverride dari parent class agar setiap penarikan mengurangi poin sesuai jumlah transaksi.

B. Latihan 2

1. Kode Program

```python
class Karyawan:
    def __init__(self):
        self.gaji_pokok = 5000000

    def hitung_bonus(self):
        return 0.05 * self.gaji_pokok

    def tampilkan_gaji(self):
        bonus = self.hitung_bonus()
        total = self.gaji_pokok + bonus
        print("\n[Karyawan Umum]")
        print(f"Gaji Pokok: Rp{self.gaji_pokok}")
        print(f"Bonus: Rp{bonus}")
        print(f"Total Gaji: Rp{total}")


class Manajer(Karyawan):
    def __init__(self):
        super().__init__()
        self.gaji_pokok = 8000000
```

```
    def hitung_bonus(self, jumlah_tim):
        return (0.1 * self.gaji_pokok) + (0.02 * jumlah_tim * self.gaji_pokok)

    def tampilkan_gaji(self, jumlah_tim):
        bonus = self.hitung_bonus(jumlah_tim)
        total = self.gaji_pokok + bonus
        print("\n[Manajer]")
        print(f"Gaji Pokok: Rp{self.gaji_pokok}")
        print(f"Bonus (Tim {jumlah_tim} orang): Rp{bonus}")
        print(f"Total Gaji: Rp{total}")


class Staf_Khusus(Karyawan):
    def __init__(self):
        super().__init__()
        self.gaji_pokok = 4000000

    def hitung_bonus(self, jumlah_kehadiran):
        return super().hitung_bonus() + (0.01 * jumlah_kehadiran *
self.gaji_pokok)

    def tampilkan_gaji(self, jumlah_kehadiran):
        bonus = self.hitung_bonus(jumlah_kehadiran)
        total = self.gaji_pokok + bonus
        print("\n[Staf Khusus]")
        print(f"Gaji Pokok: Rp{self.gaji_pokok}")
        print(f"Bonus (Kehadiran {jumlah_kehadiran} hari): Rp{bonus}")
        print(f"Total Gaji: Rp{total}")


karyawan = Karyawan()
karyawan.tampilkan_gaji()

manajer = Manajer()
manajer.tampilkan_gaji(5)

staf_khusus = Staf_Khusus()
staf_khusus.tampilkan_gaji(20)
```

2. Output

```
[Karyawan Umum]
Gaji Pokok: Rp5000000
Bonus: Rp250000.0
Total Gaji: Rp5250000.0

[Manajer]
Gaji Pokok: Rp8000000
Bonus (Tim 5 orang): Rp1600000.0
Total Gaji: Rp9600000.0

[Staf Khusus]
Gaji Pokok: Rp4000000
Bonus (Kehadiran 20 hari): Rp1000000.0
Total Gaji: Rp5000000.0
```

3. Penjelasan

Pada latihan 2 ini, saya membuat tiga class yaitu “Karyawan”, “Manajer”, dan “Staf_Khusus”. Class “Karyawan” berperan sebagai parent class yang menjadi dasar bagi class lainnya, sedangkan “Manajer” dan “Staf_Khusus” merupakan child class yang mewarisi atribut serta method dari class induk, tetapi memiliki perhitungan bonus yang berbeda sesuai peran masing-masing.

Class Karyawan memiliki satu atribut utama yaitu gaji_pokok dengan nilai awal Rp5.000.000. Class ini memiliki dua method, yaitu hitung_bonus() yang menghitung bonus sebesar 5% dari gaji pokok, dan tampilkan_gaji() yang menampilkan rincian gaji pokok, bonus, serta total gaji yang diterima.

Class Manajer merupakan turunan dari Karyawan dan memiliki gaji pokok lebih besar, yaitu Rp8.000.000. Method hitung_bonus() pada class ini dioverride untuk menyesuaikan aturan bonus manajer, di mana total bonus dihitung dari 10% gaji pokok ditambah 2% dari jumlah anggota tim yang dipimpin. Method tampilkan_gaji() menampilkan hasil perhitungan tersebut dengan parameter jumlah tim.

Class Staf_Khusus juga merupakan turunan dari Karyawan, tetapi dengan gaji pokok sebesar Rp4.000.000. Method hitung_bonus() dioverride untuk menghitung bonus berdasarkan jumlah kehadiran, yaitu bonus bawaan dari class induk ditambah 1% dari gaji pokok dikali jumlah hari hadir. Kemudian tampilkan_gaji() digunakan untuk menampilkan rincian perhitungan tersebut.

C. Tugas Rumah

1. Kode Program

```python
class Produk:
    def __init__(self, nama, harga, stok):
        self.nama = nama
        self.harga = harga
        self.stok = stok

    def hitung_diskon(self, total_harga, persentase_diskon):
        diskon = (persentase_diskon / 100) * total_harga
        harga_setelah_diskon = total_harga - diskon
        return harga_setelah_diskon
```

```
    def beli(self, jumlah):
        self.stok -= jumlah
        total_harga = jumlah * self.harga
        return total_harga

    def tampilkan_detail(self, total_harga, diskon, harga_diskon):
        print(f"Total Pembelian: Rp{total_harga}")
        print(f"Diskon: {diskon}%")
        print(f"Harga Setelah Diskon: Rp{harga_diskon}")


class Elektronik(Produk):
    def __init__(self, nama, harga, stok, garansi):
        super().__init__(nama, harga, stok)
        self.garansi = garansi

    def beli(self, jumlah):
        total_harga = super().beli(jumlah)

        if total_harga > 10000000:
            diskon = 10
        else:
            diskon = 5

        harga_diskon = super().hitung_diskon(total_harga, diskon)

        print(f"\n[Elektronik] {self.nama} (Garansi: {self.garansi})")
        super().tampilkan_detail(total_harga, diskon, harga_diskon)


class Pakaian(Produk):
    def __init__(self, nama, harga, stok, ukuran):
        super().__init__(nama, harga, stok)
        self.ukuran = ukuran

    def beli(self, jumlah):
        total_harga = super().beli(jumlah)

        if total_harga > 1000000:
            diskon = 15
        else:
            diskon = 7

        harga_diskon = super().hitung_diskon(total_harga, diskon)
        print(f"\n[Pakaian] {self.nama} (Ukuran: {self.ukuran})")
        super().tampilkan_detail(total_harga, diskon, harga_diskon)
```

```python
class Makanan(Produk):
    def __init__(self, nama, harga, stok, tanggal_kadaluarsa):
        super().__init__(nama, harga, stok)
        self.tanggal_kadaluarsa = tanggal_kadaluarsa

    def beli(self, jumlah):
        total_harga = super().beli(jumlah)

        if total_harga > 500000:
            diskon = 8
        else:
            diskon = 3

        harga_diskon = super().hitung_diskon(total_harga, diskon)
        print(f"\n[Makanan] {self.nama}")
        super().tampilkan_detail(total_harga, diskon, harga_diskon)


laptop = Elektronik("Laptop", 12000000, 5, "2 Tahun")
kaos = Pakaian("Kaos", 200000, 20, "L")
snack = Makanan("Snack", 100000, 50, "12-12-2025")

laptop.beli(1)
kaos.beli(6)
snack.beli(3)
```

2. Output

```
[Elektronik] Laptop (Garansi: 2 Tahun)
Total Pembelian: Rp12000000
Diskon: 10%
Harga Setelah Diskon: Rp10800000.0

[Pakaian] Kaos (Ukuran: L)
Total Pembelian: Rp1200000
Diskon: 15%
Harga Setelah Diskon: Rp1020000.0

[Makanan] Snack
Total Pembelian: Rp300000
Diskon: 3%
Harga Setelah Diskon: Rp291000.0
```

3. Penjelasan

Pada latihan 3 ini, saya membuat empat class yaitu “Produk”, “Elektronik”, “Pakaian”, dan “Makanan”. Class “Produk” berperan sebagai parent class yang menyimpan atribut dan method dasar untuk setiap produk, sedangkan “Elektronik”, “Pakaian”, dan “Makanan” merupakan child class yang mewarisi struktur dan perilaku dari class induk, tetapi memiliki logika diskon yang berbeda sesuai jenis produk.

Class Produk memiliki tiga atribut utama yaitu nama, harga, dan stok. Terdapat tiga method di dalamnya, yaitu hitung_diskon() yang berfungsi menghitung harga setelah diskon berdasarkan total pembelian dan persentase diskon, beli() yang mengurangi stok dan menghitung total harga pembelian, serta tampilkan_detail() yang menampilkan informasi total harga, persentase diskon, dan harga akhir setelah diskon.

Class Elektronik merupakan turunan dari class Produk dengan tambahan atribut garansi. Method beli() dioverride agar dapat menambahkan aturan diskon khusus, yaitu 10% jika total harga lebih dari Rp10.000.000 dan 5% jika sama atau di bawah nilai tersebut. Setelah perhitungan selesai, method ini menampilkan rincian produk elektronik beserta garansinya.

Class Pakaian juga merupakan turunan dari class Produk dengan atribut tambahan ukuran. Method beli() pada class ini dioverride untuk memberikan diskon 15% jika total pembelian lebih dari Rp1.000.000, dan 7% jika di bawahnya. Hasil perhitungan kemudian ditampilkan dengan format yang menampilkan ukuran pakaian.

Class Makanan menambahkan atribut tanggal_kadaluarsa dan juga melakukan overriding pada method beli(). Diskon untuk makanan ditentukan sebesar 8% jika total pembelian lebih dari Rp500.000, dan 3% jika di bawahnya. Setelah diskon diterapkan, hasilnya ditampilkan melalui method tampilkan_detail() milik class induk.

Link repository Github: https://github.com/muzaaqi/pemrograman-berorientasi-objek-praktik