

Nama: Achmad Gibran

Nim: 202412003

Mata Kuliah: Pemrograman Berorientasi Objek

Tugas Relasi

1. Perpustakaan

a. SourceCode

```
Achmad Gibran Tugas 3 Relasi > Perpustakaan.py > Buku > __init__
1 # Achmad Gibran - 202412003
2 # Tugas relasi 1
3
4 class Buku:
5     def __init__(self, judul, penulis):
6         self.judul = judul
7         self.penulis = penulis
8         self.status_dipinjam = False
9
10    def __str__(self):
11        status = "Dipinjam" if self.status_dipinjam else "Tersedia"
12        return f"{self.judul} oleh {self.penulis} ({status})"
13
14
15 class Anggota:
16     def __init__(self, nama):
17         self.nama = nama
18
19     def __str__(self):
20         return f"Anggota: {self.nama}"
21
22
23 class Peminjaman:
24     def __init__(self, anggota, buku):
25         self.anggota = anggota
26         self.buku = buku
27         self.status_dipinjam = True
28
29     def __str__(self):
30         return f"{self.anggota.nama} meminjam buku '{self.buku.judul}'"
31
32
33 class Perpustakaan:
34     def __init__(self, nama):
35         self.nama = nama
36         self.koleksi_buku = []
37         self.daftar_anggota = []
38
39
40 Achmad Gibran Tugas 3 Relasi > Perpustakaan.py > Perpustakaan
33 class Perpustakaan:
34     def __init__(self, nama):
35         self.transaksi = []
36
37     def tambah_buku(self, buku):
38         self.koleksi_buku.append(buku)
39
40     def tambah_anggota(self, anggota):
41         self.daftar_anggota.append(anggota)
42
43     def pinjam_buku(self, anggota, judul_buku):
44         for buku in self.koleksi_buku:
45             if buku.judul == judul_buku:
46                 if not buku.status_dipinjam:
47                     peminjaman = Peminjaman(anggota, buku)
48                     self.transaksi.append(peminjaman)
49                     return peminjaman
50                 else:
51                     print(f"Buku '{buku.judul}' sudah dipinjam!")
52                     return None
53         print("Buku tidak ditemukan!")
54         return None
55
56     def tampilkan_buku(self):
57         print(f"\ndaftar Buku di {self.nama}:")
58         for i, buku in enumerate(self.koleksi_buku, start=1):
59             print(f"{i}. {buku}")
60
61     def tampilkan_anggota(self):
62         print(f"\ndaftar Anggota di {self.nama}:")
63         for anggota in self.daftar_anggota:
64             print(f"-. {anggota}")
65
66     def tampilkan_transaksi(self):
67         print("\nRiwayat Peminjaman:")
68         if not self.transaksi:
69             print("Belum ada transaksi.")
```

```

73         for transaksi in self.transaksi:
74             print("-", transaksi)
75
76
77 # PENGGUNAAN
78 perpus = Perpustakaan("Perpustakaan Kota")
79
80 # Tambah buku (2 pilihan saja)
81 perpus.tambah_buku(Buku("Laskar Pelangi", "Andrea Hirata"))
82 perpus.tambah_buku(Buku("Bumi Manusia", "Pramoedya Ananta Toer"))
83
84 # Tambah anggota (Bram dan Gibran)
85 bram = Anggota("Bram")
86 gibran = Anggota("Gibran")
87 perpus.tambah_anggota(bram)
88 perpus.tambah_anggota(gibran)
89
90 # Bram otomatis pinjam buku pertama
91 perpus.pinjam_buku(bram, "Laskar Pelangi")
92
93 # Tampilkan daftar buku sebelum Gibran memilih
94 perpus.tampilkan_buku()
95
96 # Gibran memilih buku yang tersisa
97 pilihan = int(input("\nGibran, pilih nomor buku yang ingin dipinjam: "))
98
99 if pilihan in [1, 2]:
100     buku_dipilih = perpus.koleksi_buku[pilihan - 1].judul
101     perpus.pinjam_buku(gibran, buku_dipilih)
102 else:
103     print("Buku tidak tersedia.")
104
105 # Tampilkan hasil akhir
106 perpus.tampilkan_buku()
107 perpus.tampilkan_anggota()
108 perpus.tampilkan_transaksi()
109

```

b. Output

```

Daftar Buku di Perpustakaan Kota:
1. Laskar Pelangi oleh Andrea Hirata (Dipinjam)
2. Bumi Manusia oleh Pramoedya Ananta Toer (tersedia)

Gibran, pilih nomor buku yang ingin dipinjam: 2

Daftar Buku di Perpustakaan Kota:
1. Laskar Pelangi oleh Andrea Hirata (Dipinjam)
2. Bumi Manusia oleh Pramoedya Ananta Toer (Dipinjam)

Daftar Anggota di Perpustakaan Kota:
- Anggota: Bram
- Anggota: Gibran

Riwayat Peminjaman:
- Bram meminjam buku 'Laskar Pelangi'
- Gibran meminjam buku 'Bumi Manusia'

```

c. Analisis

Kelas Perpustakaan menunjukkan agregasi, di mana ia mengelola koleksi independen dari objek Buku dan Anggota (koleksi ini tetap ada meskipun perpustakaan dibubarkan). Relasi Asosiasi diimplementasikan melalui kelas perantara Peminjaman, yang secara eksplisit menghubungkan satu objek Anggota dengan satu objek Buku, sekaligus memicu mutasi state pada objek Buku dengan mengubah status_dipinjam menjadi True. Struktur ini memastikan bahwa semua aktivitas utama sistem, mulai dari penambahan entitas (tambah_buku, tambah_anggota) hingga eksekusi transaksi (pinjam_buku), dikelola secara terpusat oleh objek Perpustakaan, menjaga integritas data status peminjaman pada setiap objek Buku yang terlibat.

2. E-Commerce

a. SourceCode

```
Achmad Gibran Tugas 3 Relasi > E-Commerce.py > Pelanggan > buat_pesanan
1  # Achmad Gibran - 202412003
2  # Tugas relasi 2
3
4  class Pelanggan:
5      def __init__(self, nama, email):
6          # Implementasi: Inisialisasi pelanggan dengan keranjang dan riwayat pesanan
7          self.nama = nama
8          self.email = email
9          self.keranjang = Keranjang()
10         self.riwayat_pesanan = []
11
12     def buat_pesanan(self):
13         # Implementasi: buat pesanan dari keranjang
14         if not self.keranjang.items:
15             return None
16
17         pesanan_baru = Pesanan(self)
18
19         for produk, jumlah in self.keranjang.items.items():
20             # Proses item, buat ItemPesanan, dan update stok
21             item = ItemPesanan(produk, jumlah)
22             pesanan_baru.item_pesanan_list.append(item)
23             produk.stok -= jumlah
24
25         pesanan_baru.total_harga = sum(item.hitung_subtotal() for item in pesanan_baru.item_pesanan_list)
26         self.riwayat_pesanan.append(pesanan_baru)
27         self.keranjang.items = {} # Kosongkan keranjang
28
29         return pesanan_baru
30
31     class Produk:
32         def __init__(self, nama, harga, stok):
33             self.nama = nama
34             self.harga = harga
35             self.stok = stok
36
37     class Keranjang:
38         def __init__(self):
39             self.items = {} # {Produk: jumlah}
40
41         def tambah_produk(self, produk, jumlah):
42             if produk.stok < jumlah:
43                 return
44
45             self.items[produk] = self.items.get(produk, 0) + jumlah
46
47         # Metode hapus_keranjang dihilangkan, langsung di Kosongkan di Pelanggan.buat_pesanan
48
49     class Pesanan:
50         def __init__(self, pelanggan):
51             self.pelanggan = pelanggan
52             self.item_pesanan_list = []
53             self.total_harga = 0
54
55         # Metode hitung total digabungkan langsung ke Pelanggan.buat_pesanan untuk penyederhanaan
56
57     class ItemPesanan:
58         def __init__(self, produk, jumlah):
59
60             self.produk = produk
61             self.jumlah = jumlah
62
63         def hitung_subtotal(self):
64             return self.produk.harga * self.jumlah
65
66     # PROGRAM
67     if __name__ == "__main__":
68
69         # SETUP
70         p1 = Produk("Buku", 50000, 10)
71         p2 = Produk("Pensil", 10000, 5)
72
73         koko = Pelanggan("Gibran", "Gibran@gmail.com")
74
75         # PROSES
76         koko.keranjang.tambah_produk(p1, 2)
77         koko.keranjang.tambah_produk(p2, 1)
78
79         pesanan = koko.buat_pesanan()
80
81         # OUTPUT
82         if pesanan:
83             print(f"Pesanan dari {pesanan.pelanggan.nama} berhasil dibuat.")
84             print(f"Total Harga: Rp {pesanan.total_harga}")
85             print(f"Stok Buku sekarang: {p1.stok}")
86         else:
87             print("Gagal membuat pesanan.")
```

b. Output

```
Pesanan dari Gibran berhasil dibuat.
Total Harga: Rp 110000
Stok Buku sekarang: 8
PS C:\Users\ACER NITRO\Documents\Achmad_Gibran_Tugas>
```

c. Analisis

Pelanggan memiliki Keranjang dan Pesanan terdiri dari ItemPesanan. Sementara itu, Asosiasi diimplementasikan melalui rujukan silang yang tidak terikat kepemilikan, di mana ItemPesanan merujuk pada Produk dan Pesanan merujuk balik pada Pelanggan. Metode kunci Pelanggan.buat_pesanan bertindak sebagai controller utama yang mengelola transisi state data: dari daftar sementara di Keranjang menjadi data permanen di objek Pesanan dan riwayat_pesanan, sambil secara esensial menjalankan proses bisnis dengan mengurangi stok pada objek Produk yang terasosiasi.

3. Proyek

a. SourceCode

```
Achmad Gibran Tugas 3 Relasi > Proyek.py > Perusahaan > __init__
1  # Achmad Gibran - 202412003
2  # Tugas relasi 3
3
4  class Perusahaan:
5      def __init__(self, nama):
6          # inisialisasi perusahaan dengan daftar proyek dan tim
7          self.nama = nama
8          self.proyek_list = [] # Aggregation: Perusahaan memiliki proyek
9          self.tim_list = [] # Composition: Tim bagian dari perusahaan
10         pass
11
12     def buat_proyek(self, nama_proyek, deskripsi):
13         # buat proyek baru
14         proyek = Proyek(nama_proyek, deskripsi)
15         self.proyek_list.append(proyek)
16         return proyek
17         pass
18
19     def buat_tim(self, nama_tim):
20         # buat tim baru
21         tim = Tim(nama_tim)
22         self.tim_list.append(tim)
23         return tim
24         pass
25
26
27     class Proyek:
28         def __init__(self, nama, deskripsi):
29             # inisialisasi proyek dengan daftar tugas
30             self.nama = nama
31             self.deskripsi = deskripsi
32             self.tugas_list = [] # Composition: Tugas bagian dari proyek
33             pass
34
35         def tambah_tugas(self, deskripsi_tugas):
36             # tambahkan tugas ke proyek
37             tugas = Tugas(deskripsi_tugas)
38             self.tugas_list.append(tugas)
39             return tugas
40             pass
41
42
43     class Tim:
44         def __init__(self, nama):
45             # Implementasi: inisialisasi tim dengan daftar developer
46             self.nama = nama
47             self.developers = [] # Aggregation: Tim memiliki developer
48             pass
49
50         def tambah_developer(self, developer):
51             # Implementasi: tambahkan developer ke tim
52             self.developers.append(developer)
53             pass
54
55
56     class Developer:
57         def __init__(self, nama, keahlian):
58             # Implementasi: inisialisasi developer
59             self.nama = nama
60             self.keahlian = keahlian
61             pass
62
63
64     class Tugas:
65         def __init__(self, deskripsi):
66             # Implementasi: inisialisasi tugas
67             self.deskripsi = deskripsi
68             self.developer = None # Association: Tugas dapat ditugaskan ke developer
69             pass
70
71         def tugaskan_ke(self, developer):
72             # Implementasi: tugaskan tugas ke developer
```

```

73         self.developer = developer
74         pass
75
76
77 # =====
78 # PROGRAM UTAMA (TESTING)
79 # =====
80 if __name__ == "__main__":
81     # 1. Membuat Perusahaan
82     perusahaan = Perusahaan("Tech Innovators")
83
84     # 2. Membuat tim dan menambah developer
85     tim_backend = perusahaan.buat_tim("Backend")
86     dev1 = Developer("Bram", "Backend Developer")
87     dev2 = Developer("Zaldu", "Database Specialist")
88     tim_backend.tambah_developer(dev1)
89     tim_backend.tambah_developer(dev2)
90
91     tim_frontend = perusahaan.buat_tim("Frontend")
92     dev3 = Developer("Gibran", "Frontend Developer")
93     tim_frontend.tambah_developer(dev3)
94
95     # 3. Membuat proyek dan menambah tugas
96     proyek1 = perusahaan.buat_proyek("Sistem E-Commerce", "Membuat platform belanja online")
97     tugas1 = proyek1.tambah_tugas("Membuat API Produk")
98     tugas2 = proyek1.tambah_tugas("Mendesain Halaman Utama")
99
100    # 4. Menugaskan tugas ke developer
101    tugas1.tugaskan_ke(dev1)
102    tugas2.tugaskan_ke(dev3)
103
104    # 5. Menampilkan status proyek dan tugas
105    print(f"Perusahaan: {perusahaan.nama}")
106    print(f"\nDaftar Tim dan Developer:")
107    for tim in perusahaan.tim_list:
108        print(f"  Tim: {tim.nama}")
109        for dev in tim.developers:
110            print(f"    • {dev.nama} ({dev.keahlian})")
111
112    print("\nDaftar Proyek dan Tugas:")
113    for proyek in perusahaan.proyek_list:
114        print(f"- Proyek: {proyek.nama} ({proyek.deskripsi})")
115        for tugas in proyek.tugas_list:
116            if tugas.developer:
117                print(f"  • Tugas: {tugas.deskripsi} → Dikerjakan oleh {tugas.developer.nama}")
118            else:
119                print(f"  • Tugas: {tugas.deskripsi} → Belum ditugaskan")
120

```

b. Output

```

Daftar Proyek dan Tugas:
- Proyek: Sistem E-Commerce (Membuat platform belanja online)
  • Tugas: Membuat API Produk → Dikerjakan oleh Bram
  • Tugas: Mendesain Halaman Utama → Dikerjakan oleh Gibran

```

c. Analisis

Tim dan Proyek terikat pada Perusahaan, dan Tugas terikat pada Proyek. Sementara itu, Agregasi diimplementasikan pada level yang lebih rendah, seperti Tim yang mengelola kumpulan Developer independen. Hubungan fungsional utama diwujudkan melalui Asosiasi dinamis, khususnya saat objek Tugas menetapkan referensi ke objek Developer melalui metode `tugaskan_ke`. Desain ini menjamin aliran kerja yang terpusat dan terstruktur, di mana controller utama (Perusahaan dan Proyek) mengelola siklus hidup elemen turunan (Tim, Tugas), sementara Developer dan Proyek berpartisipasi dalam relasi yang lebih fleksibel.