LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 10

2411102441053

Radhitya Andromeda Barito

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : 2411102441053@umkt.ac.id

Kode Lengkap Sistem Perpustakaan (Tugas D)

anggota.py

sistem-perpustakaan/anggota.py

```python

class Anggota:
    def __init__(self, nama):
        self.nama = nama
        self.daftar_buku = []

    def pinjam_buku(self, buku):
        if not buku.dipinjam:
            buku.dipinjam = True
            self.daftar_buku.append(buku)
            print(f"{self.nama} meminjam buku '{buku.judul}'")
        else:
            print(f"Buku '{buku.judul}' sedang dipinjam orang lain")

    def kembalikan_buku(self, buku):
        if buku in self.daftar_buku:
            buku.dipinjam = False
            self.daftar_buku.remove(buku)
            print(f"{self.nama} mengembalikan buku '{buku.judul}'")
        else:
            print(f"{self.nama} tidak meminjam buku '{buku.judul}'")
```

buku.py

sistem-perpustakaan/buku.py

```python

class Buku:
    def __init__(self, judul, penulis):
        self.judul = judul
        self.penulis = penulis
        self.dipinjam = False

    def tampilkan_info(self):
        status = "Dipinjam" if self.dipinjam else "Tersedia"
        return f"Buku: {self.judul}, Penulis: {self.penulis}, Status: {status}"
```

EMAIL : 2411102441053@umkt.ac.id

pustaka.py

sistem-perpustakaan/pustaka.py

```python

class Pustaka:
    def __init__(self):
        self.koleksi_buku = []

    def tambah_buku(self, buku):
        self.koleksi_buku.append(buku)

    def tampilkan_koleksi(self):
        print("\n--- Koleksi Buku di Pustaka ---")
        for buku in self.koleksi_buku:
            print(buku.tampilkan_info())
```

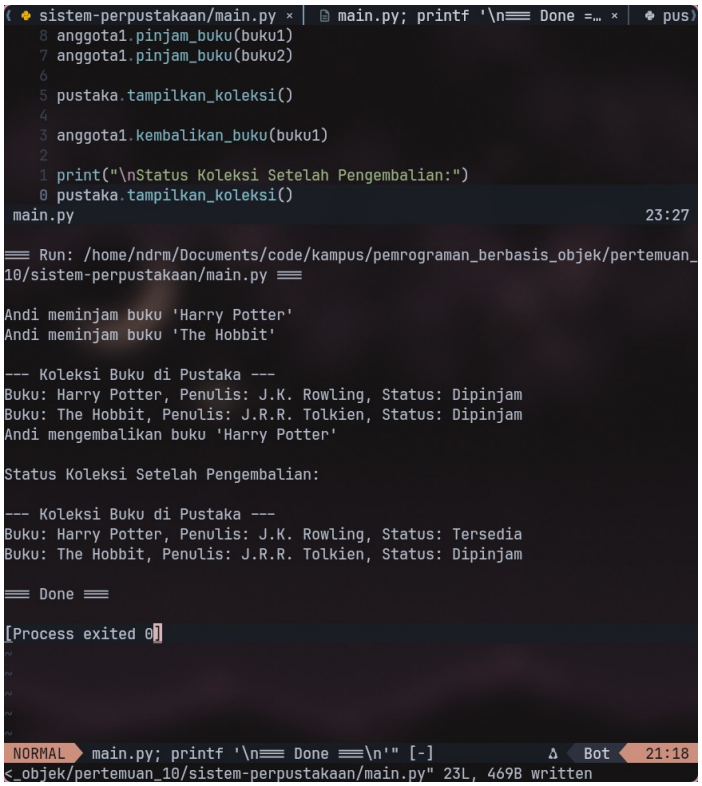
main.py

sistem-perpustakaan/main.py

```python

from buku import Buku
from pustaka import Pustaka
from anggota import Anggota

buku1 = Buku("Harry Potter", "J.K. Rowling")
buku2 = Buku("The Hobbit", "J.R.R. Tolkien")

pustaka = Pustaka()
pustaka.tambah_buku(buku1)
pustaka.tambah_buku(buku2)

anggota1 = Anggota("Andi")

anggota1.pinjam_buku(buku1)
anggota1.pinjam_buku(buku2)

pustaka.tampilkan_koleksi()

anggota1.kembalikan_buku(buku1)

print("\nStatus Koleksi Setelah Pengembalian:")
pustaka.tampilkan_koleksi()
```

EMAIL : 2411102441053@umkt.ac.id

Output



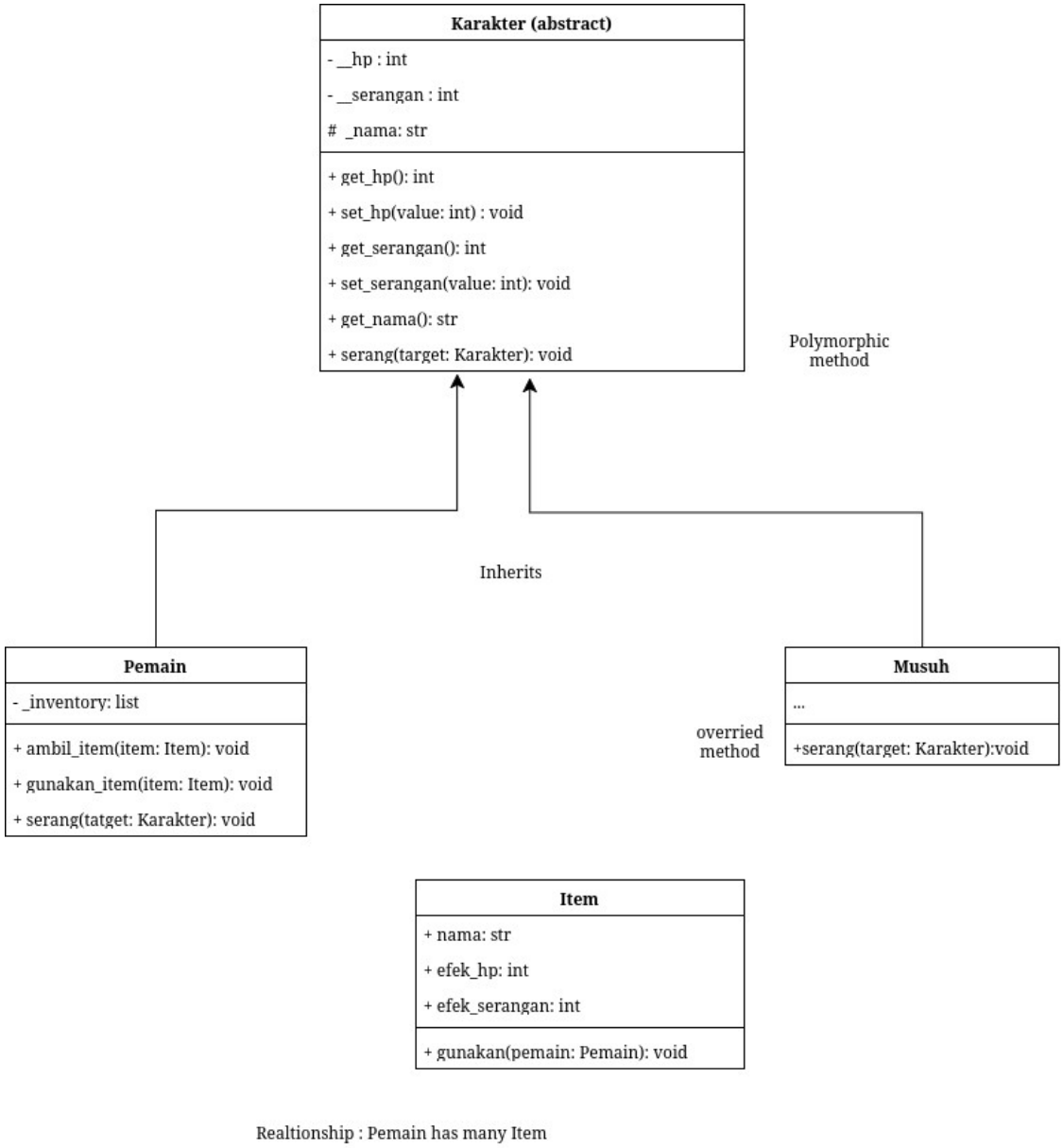
Program ini menerapkan konsep Object-Oriented Programming (OOP) dengan membagi sistem ke dalam tiga kelas utama, yaitu Buku, Pustaka, dan Anggota. Kelas Buku bertanggung jawab menyimpan data buku, seperti judul, penulis, dan status peminjaman. Kelas Pustaka berfungsi untuk mengelola koleksi buku, termasuk menambahkan buku ke dalam daftar. Sementara itu, kelas Anggota digunakan untuk merepresentasikan pengguna perpustakaan yang dapat melakukan tindakan meminjam dan mengembalikan buku. Dengan pembagian kelas seperti ini, program menjadi modular, lebih terstruktur, serta mudah dikembangkan.

Alur program dimulai dari file main.py, yang berperan sebagai titik eksekusi utama. Pada tahap awal, objek buku dibuat dan ditambahkan ke dalam pustaka. Selanjutnya, objek anggota melakukan peminjaman dengan memanggil method pinjam_buku(), yang secara otomatis mengubah status buku menjadi dipinjam. Program kemudian menampilkan daftar koleksi buku beserta statusnya melalui method tampilkan_koleksi(). Setelah itu, buku yang telah dipinjam dapat dikembalikan menggunakan method kembalikan_buku(), dan status buku berubah kembali menjadi tersedia. Interaksi antar objek ini menunjukkan proses komunikasi antar kelas, di mana perubahan status pada satu objek dipengaruhi oleh aksi dari objek lain.

EMAIL : 2411102441053@umkt.ac.id

Sistem game RPG (Tugas E)

UML (Unified Modeling Language)



EMAIL : 2411102441053@umkt.ac.id

karakter.py

game-rpg/entities/karakter.py

```python

class Karakter:
    def __init__(self, nama, hp, serangan):
        self._nama = nama
        self.__hp = hp
        self.__serangan = serangan

    def get_hp(self):
        return self.__hp

    def get_serangan(self):
        return self.__serangan

    def get_nama(self):
        return self._nama

    def set_hp(self, value):
        if value < 0:
            self.__hp = 0
        else:
            self.__hp = value

    def set_serangan(self, value):
        if value < 0:
            self.__serangan = 0
        else:
            self.__serangan = value

    def serang(self, target):
        """akan di-override oleh child (polymorphism)"""
        pass
```

musuh.py

game-rpg/entities/musuh.py

```python

from .karakter import Karakter

class Musuh(Karakter):
    def __init__(self, nama, hp, serangan):
        super().__init__(nama, hp, serangan)

    def serang(self, target):
        print(f"{self.get_nama()} menyerang {target.get_nama()} dengan ganas!")
        target.set_hp(target.get_hp() - self.get_serangan())
        print(f"{target.get_nama()} kehilangan {self.get_serangan()} HP! Sisa HP: {target.get_hp()}")
```

EMAIL : 2411102441053@umkt.ac.id

pemain.py

game-rpg/entities/pemain.py

```python

from .karakter import Karakter

class Pemain(Karakter):
    def __init__(self, nama, hp, serangan):
        super().__init__(nama, hp, serangan)
        self._inventory = []

    def ambil_item(self, item):
        self._inventory.append(item)
        print(f"{self.get_nama()} mengambil item: {item.nama}")

    def gunakan_item(self, item):
        if item in self._inventory:
            item.gunakan(self)
            self._inventory.remove(item)
        else:
            print("Item tidak ditemukan.")

    def serang(self, target):
        print(f"{self.get_nama()} menyerang {target.get_nama()}!")
        target.set_hp(target.get_hp() - self.get_serangan())
        print(f"{target.get_nama()} menerima {self.get_serangan()} damage. Sisa HP: {target.get_hp()}")
```

item.py

game-rpg/items/item.py

```python

class Item:
    def __init__(self, nama, efek_hp=0, efek_serangan=0):
        self.nama = nama
        self.efek_hp = efek_hp
        self.efek_serangan = efek_serangan

    def gunakan(self, pemain):
        pemain.set_hp(pemain.get_hp() + self.efek_hp)
        pemain.set_serangan(pemain.get_serangan() + self.efek_serangan)
        print(f"{pemain.get_nama()} menggunakan {self.nama}! (+{self.efek_hp} HP, +{self.efek_serangan} Serangan)")
```

EMAIL : 2411102441053@umkt.ac.id

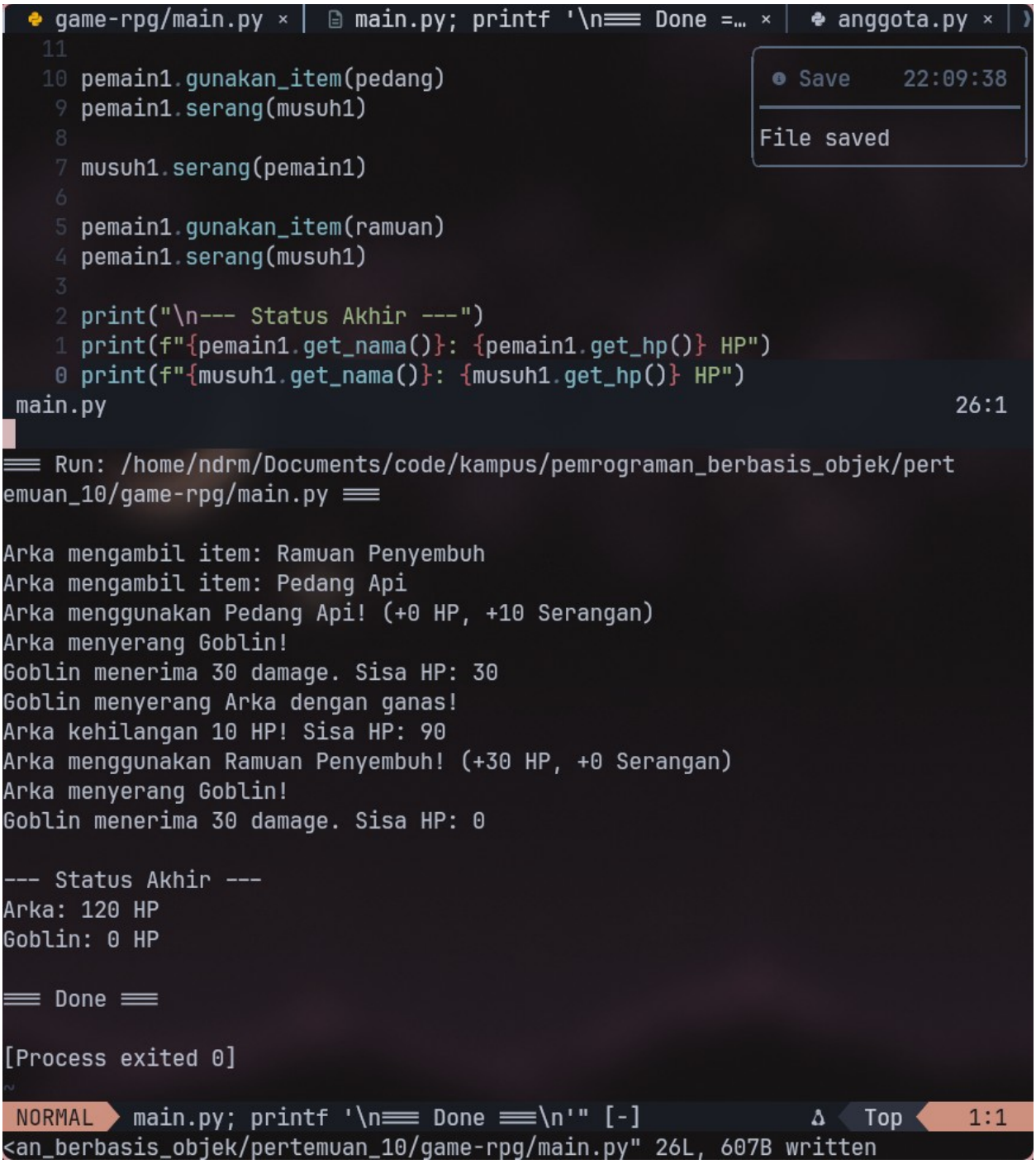
main.py

game-rpg/main.py

```
1
2
3  from entities.pemain import Pemain
4  from entities.musuh import Musuh
5  from items.item import Item
6
7  pemain1 = Pemain("Arka", hp=100, serangan=20)
8  musuh1 = Musuh("Goblin", hp=60, serangan=10)
9
10 ramuan = Item("Ramuan Penyembuh", efek_hp=30)
11 pedang = Item("Pedang Api", efek_serangan=10)
12
13 pemain1.ambil_item(ramuan)
14 pemain1.ambil_item(pedang)
15
16 pemain1.gunakan_item(pedang)
17 pemain1.serang(musuh1)
18
19 musuh1.serang(pemain1)
20
21 pemain1.gunakan_item(ramuan)
22 pemain1.serang(musuh1)
23
24 print("\n--- Status Akhir ---")
25 print(f"{pemain1.get_nama()}: {pemain1.get_hp()} HP")
26 print(f"{musuh1.get_nama()}: {musuh1.get_hp()} HP")
```

EMAIL : 2411102441053@umkt.ac.id

Output



EMAIL : 2411102441053@umkt.ac.id

Program ini menerapkan konsep pemrograman berorientasi objek (OOP) dengan struktur modular menggunakan beberapa file terpisah dalam folder entities dan items. Class Karakter berperan sebagai parent class yang mendefinisikan atribut dasar berupa nama, HP, dan serangan, serta menyediakan method getter dan setter untuk menerapkan prinsip enkapsulasi melalui atribut privat. Class Pemain dan Musuh merupakan child class yang mewarisi sifat dari class Karakter menggunakan mekanisme inheritance melalui super(). Kedua class tersebut melakukan overriding terhadap method serang(), sehingga menampilkan perilaku berbeda meskipun memiliki nama method yang sama, yang menunjukkan penerapan polymorphism. Selain itu, terdapat class Item yang digunakan untuk memberi efek perubahan nilai HP atau serangan pada objek pemain melalui method gunakan().

Program dimulai dari file main.py, dimana objek pemain dan musuh dibentuk menggunakan class Pemain dan Musuh. Selanjutnya dibuat beberapa instance item dan dimasukkan ke dalam inventory pemain menggunakan method ambil_item(). Pemain kemudian dapat menggunakan item tersebut melalui method gunakan_item(), yang memanggil method gunakan() untuk memberikan efek peningkatan atribut sesuai item yang digunakan. Setelah itu terjadi proses pertarungan: pemain dan musuh saling menyerang dengan memanggil method serang(), yang mengurangi nilai HP lawan berdasarkan serangan masing-masing. Di akhir program, status HP pemain dan musuh ditampilkan sebagai output akhir. Alur ini menunjukkan interaksi antar objek yang saling berkomunikasi dan memperlihatkan penerapan OOP secara utuh.

Dalam program ini juga terdapat penerapan hubungan antar objek dalam OOP.

a. Asosiasi (Association)
Asosiasi terjadi antara objek Pemain dan Musuh. Keduanya saling berinteraksi melalui pemanggilan method serang(), namun tidak saling memiliki atau menyimpan objek satu sama lain secara permanen.

b. Agregasi (Aggregation)
Agregasi terjadi antara Pemain dan Item. Pemain memiliki inventory berupa list yang menyimpan objek Item, tetapi Item tetap dapat ada tanpa pemain. Artinya hubungan “has-a”, namun tidak bergantung sepenuhnya.

c. Komposisi (Composition)
Komposisi terlihat dalam struktur program dimana class Pemain, Musuh, dan Karakter membentuk bagian dari sistem permainan. Tanpa class-class ini, program tidak dapat berjalan, sehingga hubungan ini menunjukkan ketergantungan penuh.

EMAIL : 2411102441053@umkt.ac.id