LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK (PBO) – [TUGAS 02]

Disusun Oleh

Nayla Devina Febrianti 123140061

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SUMATERA

2025

Soal Tugas

- Minggu ini hanya terdiri dari 1 Problem Set. Kalian perlu membuat sebuah permainan sederhana tentang pertarungan Robot.
    - Kalian akan membuat kelas Robot yang terdiri dari beberapa properti seperti attack, Hp, dll., serta beberapa metode seperti attack_enemy() atau regen_health().
    - Permainan ini akan berakhir ketika salah satu robot memiliki Hp = 0.
- Kalian bisa lebih kreatif dengan menambahkan konsep seperti attack_accuracy agar serangan dapat meleset dalam beberapa kesempatan atau menambahkan mekanisme skill seperti stun, silence, dll., pada musuh. (Bagian ini opsional).
- Kalian mungkin perlu dua kelas:
    - Kelas Robot: Berisi mekanisme attack, hp, dan mekanisme pertarungan robot.
    - Kelas Game: Berfungsi untuk menentukan jumlah ronde serta mengatur jalannya permainan.
- Kalian bisa melihat contoh cara kerja program di bawah ini. ini cuma contoh, kalian bisa buat versi kalian sendiri, yang lebih sederhana atau kompleks, tergantung kreativitas kalian.

Penjelasan code :

Struktur Permainan

Di dalam permainan ini, kita memiliki dua kelas utama: **Robot** dan **Game**.

1. **Kelas Robot**:
    - **Tujuan**: Menggambarkan karakter robot dalam permainan.
    - **Atribut**:
        - name: Nama robot.
        - hp: Kesehatan (Hit Points) robot, yang berkurang jika diserang.
        - attack: Daya serang robot, yang menentukan seberapa besar damage yang dapat diberikan ke musuh.
        - defense: Pertahanan robot, yang mengurangi damage yang diterima saat diserang.
        - accuracy: Akurasi serangan robot, yang menentukan seberapa besar peluang serangannya berhasil.
    - **Metode**:
        - attack_enemy(enemy): Fungsi untuk menyerang musuh. Serangan hanya berhasil jika acakannya sesuai dengan akurasi robot. Kerusakan yang diterima musuh juga dipengaruhi oleh pertahanan musuh.
        - defend(): Fungsi ini memberi tahu bahwa robot memilih untuk bertahan. Meskipun dalam implementasi ini bertahan tidak mempengaruhi gameplay, biasanya bertahan akan mengurangi kerusakan yang diterima.
        - is_alive(): Fungsi untuk memeriksa apakah robot masih hidup (apakah HP-nya lebih dari 0).

        - status(): Fungsi untuk menampilkan status robot saat ini (nama, HP, serangan, pertahanan).
2. **Kelas Game**:
    - **Tujuan**: Mengelola seluruh jalannya permainan.
    - **Metode**:
        - print_status(): Fungsi ini menampilkan status kedua robot pada awal setiap ronde. Menampilkan HP dan atribut lainnya.
        - player_action(robot): Fungsi yang meminta pemain memilih aksi untuk robot mereka. Pemain bisa memilih antara **Attack** (serang), **Defense** (bertahan), atau **Giveup** (menyerah).
        - play_round(): Fungsi untuk melaksanakan satu ronde permainan. Di sini, kedua robot memilih aksi mereka dan saling menyerang atau bertahan.
        - start(): Fungsi untuk memulai permainan dan mengatur alur permainan, memeriksa apakah ada robot yang kalah dan menentukan pemenang.

Cara Kerja Permainan:

1. **Inisialisasi**:
    - Kita buat dua robot: **Atreus** dan **Daedalus**. Masing-masing memiliki atribut seperti HP, serangan, pertahanan, dan akurasi serangan.
2. **Ronde Permainan**:
    - Setiap ronde, kedua robot akan memilih aksi. Pemain memilih antara:
        - **Serang**: Robot menyerang musuh. Serangan berhasil atau gagal tergantung pada akurasi robot dan pertahanan musuh.
        - **Bertahan**: Robot mencoba mengurangi kerusakan yang diterimanya.
        - **Menyerah**: Robot menyerah dan kalah, mengakhiri permainan.
3. **Serangan dan Pertahanan**:
    - Jika robot memilih untuk menyerang, ada kemungkinan serangannya berhasil atau meleset, tergantung pada **accuracy** yang dimiliki robot.
    - Setiap kali serangan terjadi, damage yang diterima oleh musuh akan dikurangi oleh nilai defense-nya. Misalnya, jika robot A menyerang robot B dan damage yang dihitung adalah 10, dan robot B memiliki defense 5, maka B hanya akan menerima 5 damage.
4. **Mengakhiri Permainan**:
    - Permainan berlanjut hingga salah satu robot kehabisan HP atau memilih untuk menyerah. Robot yang memiliki HP lebih tinggi di akhir permainan akan menang.

Contoh Jalannya Permainan:

Misalnya, berikut adalah apa yang terjadi selama permainan:

- Pada ronde pertama, Atreus menyerang Daedalus dan berhasil memberikan kerusakan. Namun, Daedalus memutuskan untuk bertahan, yang mengurangi kerusakan yang diterimanya.
- Pada ronde kedua, Atreus memilih untuk menyerah, dan permainan berakhir dengan Daedalus menang.

Rangkuman Langkah-Langkah Permainan:

1. Setiap ronde, kedua robot akan memilih aksi mereka, apakah menyerang, bertahan, atau menyerah.
2. Serangan dihitung berdasarkan akurasi dan pertahanan, dan HP robot akan berkurang jika diserang.
3. Permainan berlanjut hingga salah satu robot menyerah atau kehabisan HP.
4. Pemenang adalah robot yang masih hidup di akhir permainan.

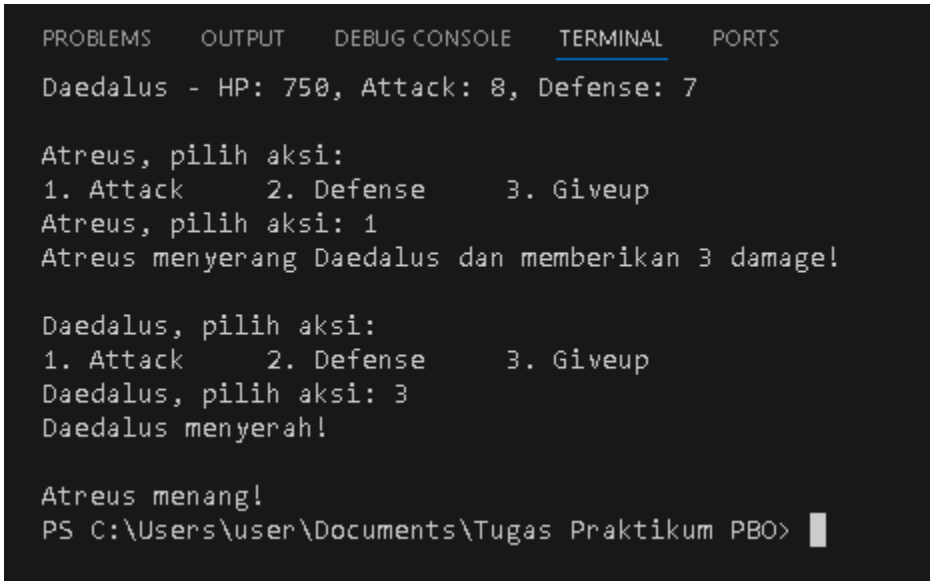
Penjelasan Fitur:

- **Accuracy**: Menentukan seberapa besar peluang serangan robot bisa mengenai musuh. Jika angka acak di bawah atau sama dengan nilai akurasi, serangan akan berhasil.
- **Defense**: Menentukan seberapa banyak kerusakan yang dikurangi oleh robot saat diserang.
- **Giveup**: Salah satu pilihan yang memungkinkan robot untuk menyerah dan kalah secara otomatis.

Source Code :

```python
import random

class Robot:
    def __init__(self, name, hp, attack, defense, accuracy):
        self.name = name
        self.hp = hp
        self.attack = attack
        self.defense = defense
        self.accuracy = accuracy

    def attack_enemy(self, enemy):
        if random.random() <= self.accuracy:
            damage = self.attack - enemy.defense
            damage = max(damage, 0)
            print(f"{self.name} menyerang {enemy.name} dan memberikan {damage} damage!")
            enemy.hp -= damage
        else:
            print(f"{self.name} gagal menyerang {enemy.name}!")

    def defend(self):
        print(f"{self.name} bertahan dengan meningkatkan pertahanan!")

    def is_alive(self):
        return self.hp > 0

    def status(self):
        print(f"{self.name} - HP: {self.hp}, Attack: {self.attack}, Defense: {self.defense}")


class Game:
    def __init__(self, robot1, robot2):
        self.robot1 = robot1
        self.robot2 = robot2
        self.round = 1

    def print_status(self):
        print(f"Round-{self.round} ==========================================================")
        self.robot1.status()
        self.robot2.status()

    def player_action(self, robot):
        print(f"\n{robot.name}, pilih aksi:")
        print("1. Attack     2. Defense     3. Giveup")
        choice = input(f"{robot.name}, pilih aksi: ")

        if choice == "1":
            return "attack"
        elif choice == "2":
            return "defend"
        elif choice == "3":
            return "giveup"
        else:
            print("Pilihan tidak valid, pilih 1, 2, atau 3.")
            return self.player_action(robot)

    def play_round(self):
        self.print_status()

        action1 = self.player_action(self.robot1)
        if action1 == "attack":
            self.robot1.attack_enemy(self.robot2)
        elif action1 == "defend":
            self.robot1.defend()
        elif action1 == "giveup":
            print(f"{self.robot1.name} menyerah!")
```

Output Hasil (Screenshot) :



Lampiran

1. **Link Percakapan LLM**
2. **Web referensi**