

LAPORAN PRAKTIKUM WEEK 9
PEMROGRAMAN BERORIENTASI OBJEK
“MEMBUAUT GAME SEDERHANA”



Oleh:
Benony Gabriel
105222002

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS SAINS DAN ILMU KOMPUTER
UNIVERSITAS PERTAMINA
2024

BAB I

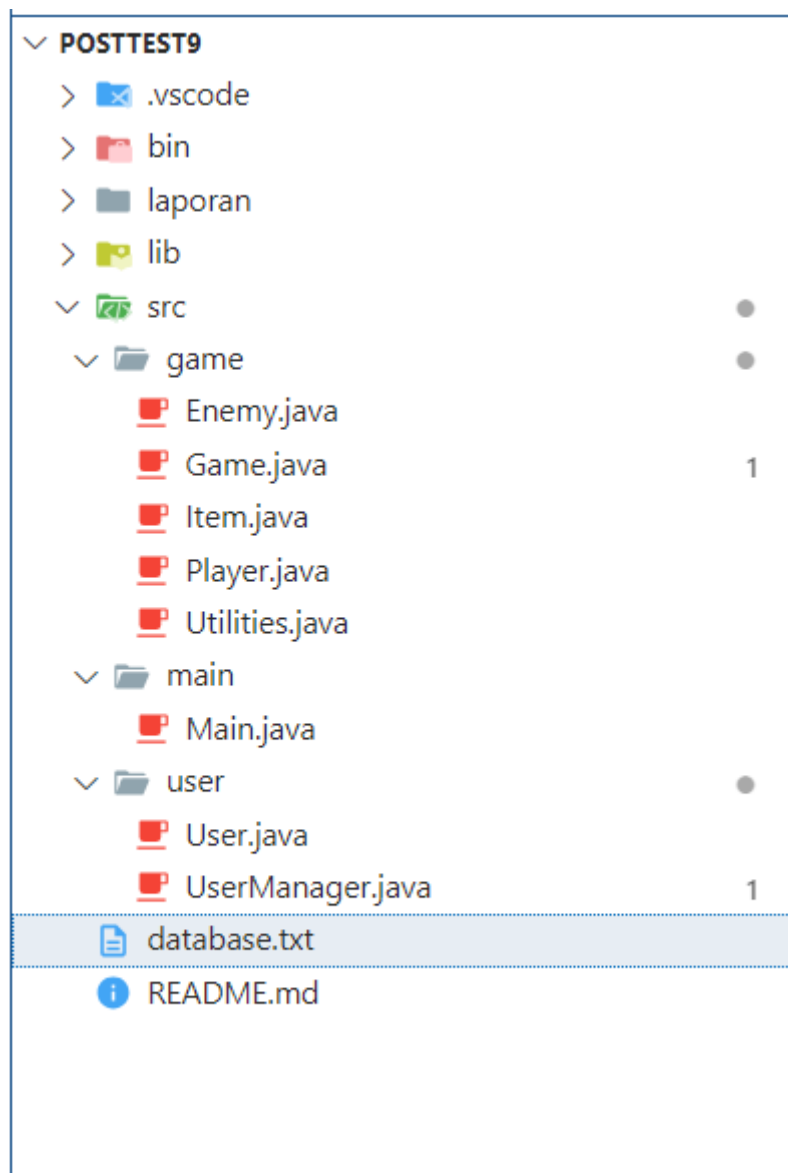
PENDAHULUAN

1.1 Studi Kasus

Dengan imajinasi dan kreasi kamu buatlah sebuah game petualangan, dimana user dapat login dan register dengan username dan password. Model game adalah game open world dimana pemain dapat maju atau mundur. Setiap Langkah akan mendapatkan sesuatu yang random baik mendapatkan item baru, musuh, ataupun yang lain(sesuaikan dengan imajinasimu). (*Studi kasus dari modul*)

1.2 Struktur Folder

Berikut ini adalah struktur folder dari game yang dikembangkan:



Gambar 1: Struktur Folder

1.3 Variabel dan Tipe Data

Untuk membuat game petualangan dengan fitur login dan register, kita akan membuat beberapa kelas di dalam package yang sesuai. Kita akan menggunakan file untuk menyimpan data user sehingga mereka bisa login kembali di lain waktu. Berikut adalah daftar variabel yang digunakan dalam game ini beserta tipe datanya dan keterangannya:

Nama Variabel	Tipe Data	Keterangan
username	String	Untuk menyimpan username pengguna
password	String	Untuk menyimpan password pengguna
health	Double	Menyimpan kesehatan pemain
mana	Double	Menyimpan mana pemain
backpack	List<String>	Menyimpan daftar item dalam tas pemain
level	Double	Menyimpan level pemain
role	String	Menyimpan peran pemain (Magician atau Fighter)
enemyName	String	Menyimpan nama musuh yang ditemukan
enemyHealth	Double	Menyimpan kesehatan musuh
enemyMana	Double	Menyimpan mana musuh
enemyAttackPower	Double	Menyimpan kekuatan serangan musuh
item	String	Menyimpan item yang ditemukan selama petualangan

Tabel 1 : Variabel dan tipe data

1.4 Constructor dan Method

Berikut adalah daftar metode yang digunakan dalam game ini, beserta jenis dan keterangannya:

1.4.1 Class User

Nama Method	Jenis	Keterangan
User	Constructor	Konstruktor untuk membuat objek User dengan username dan password
getUsername	Function	Mengembalikan username pengguna
getPassword	Function	Mengembalikan password pengguna
saveUser	Function	Menyimpan data pengguna baru ke file database
loadUsers	Function	Membaca data pengguna dari file database

Tabel 2: Method Class User

1.4.2 Class UserManager

Nama Method	Jenis	Keterangan
UserManager	Constructor	Konstruktor untuk membuat objek UserManager dan memuat pengguna dari file
register	Function	Mendaftarkan pengguna baru dan menyimpan ke file database
login	Function	Memeriksa kredensial pengguna dan mengembalikan objek User jika valid

Tabel 3: Method Class UserManager

1.4.3 Class Player

Nama Method	Jenis	Keterangan
Player	Constructor	Konstruktor untuk membuat objek Player dengan role tertentu
showBackpack	Procedure	Menampilkan item di dalam tas punggung pemain
usePotion	Procedure	Menggunakan potion untuk menambah health dan mana
addItem	Procedure	Menambahkan item ke dalam tas punggung pemain
reduceHealth	Procedure	Mengurangi health pemain dengan nilai tertentu
levelUp	Procedure	Menaikkan level pemain dan menambah health serta mana
showStatus	Procedure	Menampilkan status pemain (health, mana, level)
getHealth	Function	Mengembalikan nilai health pemain
getLevel	Function	Mengembalikan nilai level pemain

Tabel 4: Method Class Player

1.4.4 Class Enemy

Nama Method	Jenis	Keterangan
Enemy	Constructor	Konstruktor untuk membuat objek Enemy dengan atribut tertentu
getName	Function	Mengembalikan nama musuh
getHealth	Function	Mengembalikan nilai health musuh
getAttackPower	Function	Mengembalikan nilai attack power musuh

reduceHealth	Procedure	Mengurangi health musuh dengan nilai tertentu
--------------	-----------	---

Tabel 5: Method Class Enemy

1.4.5 Class Utilities

Nama Method	Jenis	Keterangan
getRandomItem	Function	Menghasilkan item acak dari daftar item
getRandomEnemy	Function	Menghasilkan musuh acak dari daftar musuh

Tabel 6: Method Class Utilities

1.4.6 Class Game

Nama Method	Jenis	Keterangan
Game	Constructor	Konstruktor untuk membuat objek Game dengan user saat ini
start	Procedure	Memulai permainan dan memungkinkan pengguna untuk memilih role dan melakukan berbagai tindakan
startAdventure	Procedure	Memulai petualangan di mana pemain dapat bergerak dan menemukan musuh atau item
battle	Procedure	Mengelola logika pertempuran antara pemain dan musuh

Tabel 7: Method Class Game

1.4.7 Class Main

Nama Method	Jenis	Keterangan
main	Function	Fungsi utama untuk menjalankan aplikasi, mengelola login, registrasi, dan memulai permainan

Tabel 8: Method Class Game

Setiap method dalam daftar di atas memainkan peran penting dalam mengelola data pengguna, logika permainan, dan interaksi antara pemain dan musuh.

BAB II

DOKUMENTASI dan PEMBAHASAN CODE

Pada bagian ini kita akan membahas lebih jauh terkait setiap method yang memainkan peran penting dalam mengelola data pengguna, logika permainan, dan interaksi antara pemain dan musuh. Kita akan melihat bagaimana method-method tersebut diimplementasikan untuk menangani registrasi, login, dan penyimpanan data pengguna. Selain itu, kita akan memeriksa logika permainan yang mengatur bagaimana pemain dapat bergerak, menemukan item, atau menghadapi musuh selama petualangan.

2.1 Class User

Kelas `User` dalam paket `user` mengelola informasi pengguna dan menyimpan serta memuat data pengguna dari file. Kelas ini memiliki dua atribut: `username` dan `password`, yang diinisialisasi melalui konstruktor. Ada dua metode utama: `saveUser(User user)` untuk menyimpan data pengguna ke file `database.txt` dan `loadUsers()` untuk memuat data pengguna dari file tersebut ke dalam sebuah map (`HashMap`). Metode `saveUser` menulis username dan password ke file, sementara metode `loadUsers` membaca data dari file dan mengembalikan map dengan username sebagai kunci dan objek `User` sebagai nilai.

2.1.1 Method saveUser

A screenshot of a code editor window titled "User.java". The code defines a static method `saveUser` that takes a `User` object as a parameter. It attempts to write the user's username and password to a file named "database.txt" using a `BufferedWriter` wrapped around a `FileWriter` in append mode. The data is formatted as "username,password". A try-catch block handles any `IOException` by printing the stack trace.

```
public static void saveUser(User user) {  
    try (BufferedWriter writer = new BufferedWriter(new FileWriter("database.txt", true))) {  
        writer.write(user.getUsername() + "," + user.getPassword());  
        writer.newLine();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

Metode `saveUser` adalah metode statis yang menyimpan data pengguna ke file `database.txt`. Metode ini menerima objek `User` sebagai parameter. Dalam metode ini, `BufferedWriter` digunakan untuk menulis data ke file. `BufferedWriter` dibungkus dengan `FileWriter` dalam mode append (`true`), sehingga data baru ditambahkan ke akhir file tanpa menghapus data yang ada. Metode ini menulis username dan password pengguna ke file dalam format "username,password" dan menambahkan baris baru setelahnya. Jika terjadi kesalahan saat menulis ke file, pengecualian `IOException` ditangkap dan ditangani dengan mencetak stack trace kesalahan.

2.1.2 Method loadUsers

A screenshot of a Java code editor window titled 'User.java'. The code defines a static method 'loadUsers()' that reads user data from a file named 'database.txt'. The method uses a 'BufferedReader' to read the file line by line. Each line is split by a comma to separate the username and password. A new 'User' object is created for each pair and added to a 'HashMap' named 'users'. The method also includes a 'try-catch' block to handle 'IOException' and a 'return' statement at the end.

```
public static Map<String, User> loadUsers() {  
    Map<String, User> users = new HashMap<>();  
    try (BufferedReader reader = new BufferedReader(new FileReader("database.txt"))) {  
        String line;  
        while ((line = reader.readLine()) != null) {  
            String[] parts = line.split(",");  
            if (parts.length == 2) {  
                users.put(parts[0], new User(parts[0], parts[1]));  
            }  
        }  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
    return users;  
}
```

Metode 'loadUsers' adalah metode statis yang memuat data pengguna dari file 'database.txt' dan mengembalikannya dalam bentuk 'Map<String, User>', dengan kunci berupa username dan nilai berupa objek 'User'. Metode ini menggunakan 'BufferedReader' untuk membaca file baris demi baris. Setiap baris yang dibaca dipisahkan oleh tanda koma untuk mendapatkan username dan password. Jika baris yang dipisahkan memiliki dua bagian (username dan password), pengguna baru dibuat dan ditambahkan ke peta. Jika terjadi kesalahan saat membaca file, pengecualian 'IOException' ditangkap dan ditangani dengan mencetak stack trace kesalahan. Metode ini mengembalikan peta yang berisi semua pengguna yang dimuat dari file.

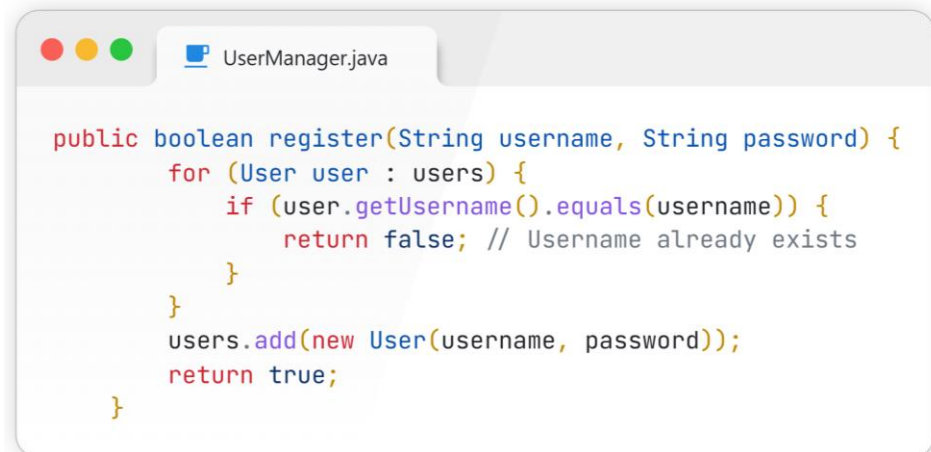
2.2 Class UserManager

Kelas 'UserManager' adalah kelas yang mengelola pendaftaran dan login pengguna. Ini menggunakan daftar 'ArrayList<User>' untuk menyimpan objek 'User'. Konstruktornya menginisialisasi daftar pengguna kosong. Metode 'register' menerima username dan password, kemudian memeriksa apakah username sudah ada dalam daftar. Jika sudah ada, pendaftaran gagal dan mengembalikan 'false'. Jika belum ada, membuat pengguna baru dan menambahkannya ke daftar, mengembalikan 'true'. Metode 'login' menerima username dan password, memeriksa setiap pengguna dalam daftar untuk mencocokkan username dan password, dan mengembalikan objek 'User' yang cocok jika ditemukan. Jika tidak ada yang cocok, mengembalikan 'null', menandakan login tidak valid.

2.2.1 Method register

Metode 'register' dalam kelas 'UserManager' digunakan untuk mendaftarkan pengguna baru dengan username dan password. Pertama, metode ini memeriksa apakah username yang diberikan sudah ada dalam daftar 'users'. Jika username sudah ada, metode mengembalikan 'false', menandakan pendaftaran gagal. Jika tidak ada pengguna dengan username yang sama, metode membuat objek 'User' baru dengan username dan

password tersebut, menambahkannya ke dalam daftar `users`, dan mengembalikan `true`, menandakan pendaftaran berhasil.



```
public boolean register(String username, String password) {
    for (User user : users) {
        if (user.getUsername().equals(username)) {
            return false; // Username already exists
        }
    }
    users.add(new User(username, password));
    return true;
}
```

Dengan demikian, metode ini memastikan setiap username unik sebelum menambahkan pengguna baru.

2.2.2 Method login



```
public User login(String username, String password) {
    for (User user : users) {
        if (user.getUsername().equals(username) && user.validatePassword(password)) {
            return user;
        }
    }
    return null; // Invalid login
}
```

Metode `login` dalam kelas `UserManager` digunakan untuk memverifikasi kredensial login pengguna. Metode ini memeriksa setiap objek `User` dalam daftar `users` untuk menemukan kecocokan dengan username dan password yang diberikan. Jika ditemukan pengguna dengan username yang sesuai dan password yang valid, metode mengembalikan objek `User` tersebut. Jika tidak ditemukan kecocokan, metode mengembalikan `null`, menandakan login tidak valid.

2.3 Class Player

Kelas `Player` dalam package `game` digunakan untuk mewakili pemain dalam permainan. Kelas ini memiliki atribut seperti `role`, `health`, `mana`, `backpack`, dan `level`. Konstruktor `Player` menginisialisasi atribut-atribut ini berdasarkan peran pemain, yaitu "Magician" atau "Fighter". Metode `usePotion` digunakan untuk meningkatkan kesehatan dan mana pemain jika memiliki potion di backpack. Metode `addItem` menambahkan item ke dalam backpack. Metode `showStatus` dan `showBackpack` menampilkan status kesehatan, mana, level, dan isi backpack pemain. Getter untuk atribut `role`, `health`, `mana`, dan `level`

memungkinkan pengambilan nilai-nilai tersebut, sementara `reduceHealth` mengurangi kesehatan pemain dan `levelUp` meningkatkan level, kesehatan, dan mana pemain.

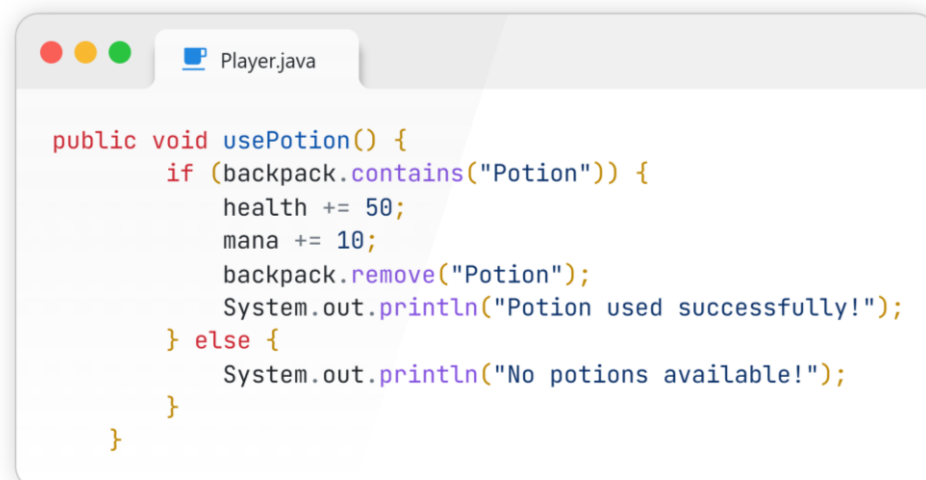
2.3.1 Constructor Player



```
public Player(String role) {
    this.role = role;
    this.health = role.equals("Magician") ? 100 : 150;
    this.mana = role.equals("Magician") ? 200 : 50;
    this.backpack = new ArrayList<>();
    this.level = 1;
}
```

Konstruktor `Player` menginisialisasi objek pemain dengan peran yang diberikan (Magician atau Fighter). Berdasarkan peran, nilai kesehatan dan mana pemain diatur. Tas punggung pemain diinisialisasi sebagai daftar kosong, dan level pemain diatur ke 1.

2.3.2 Method usePotion



```
public void usePotion() {
    if (backpack.contains("Potion")) {
        health += 50;
        mana += 10;
        backpack.remove("Potion");
        System.out.println("Potion used successfully!");
    } else {
        System.out.println("No potions available!");
    }
}
```

Metode `usePotion` mengecek apakah backpack pemain berisi potion. Jika ada, metode ini menambahkan 50 poin ke health dan 10 poin ke mana, kemudian menghapus potion dari backpack dan menampilkan pesan keberhasilan. Jika tidak ada potion, metode ini menampilkan pesan bahwa tidak ada potion yang tersedia.

2.3.3 Method useItem

A screenshot of a code editor window titled 'Player.java'. The code defines a public void method named 'addItem' that takes a 'String item' as a parameter. The method body consists of a single line: 'backpack.add(item);', followed by a closing curly brace '}'.

```
public void addItem(String item) {  
    backpack.add(item);  
}
```

Metode addItem menambahkan item yang diberikan sebagai parameter ke dalam backpack pemain.

2.3.4 Method levelUp

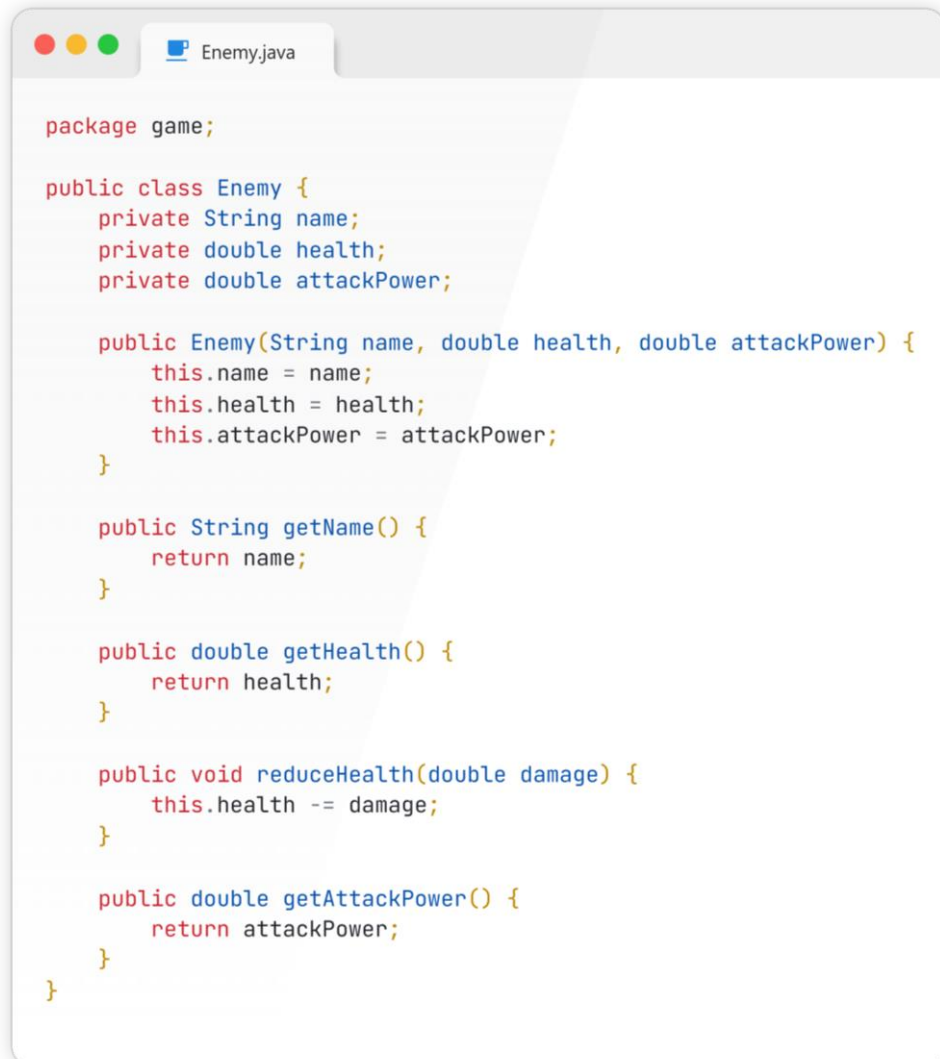
A screenshot of a code editor window titled 'Player.java'. The code defines a public void method named 'levelUp'. The method body contains three lines of code: 'this.level++;', 'this.health += 50;', and 'this.mana += 20;', followed by a closing curly brace '}'.

```
public void levelUp() {  
    this.level++;  
    this.health += 50;  
    this.mana += 20;  
}
```

Metode 'levelUp' pada kelas 'Player' meningkatkan level pemain sebanyak satu unit, serta menambah poin kesehatan pemain sebesar 50 dan poin mana sebesar 20. Kode ini bertujuan untuk memperkuat karakter pemain setiap kali mereka naik level.

2. 4 Class Enemy

Kelas 'Enemy' mendefinisikan karakter musuh dalam game dengan atribut nama, kesehatan, dan kekuatan serangan. Konstruktor kelas ini menginisialisasi atribut-atribut tersebut dengan nilai yang diberikan saat pembuatan objek.



```
package game;

public class Enemy {
    private String name;
    private double health;
    private double attackPower;

    public Enemy(String name, double health, double attackPower) {
        this.name = name;
        this.health = health;
        this.attackPower = attackPower;
    }

    public String getName() {
        return name;
    }

    public double getHealth() {
        return health;
    }

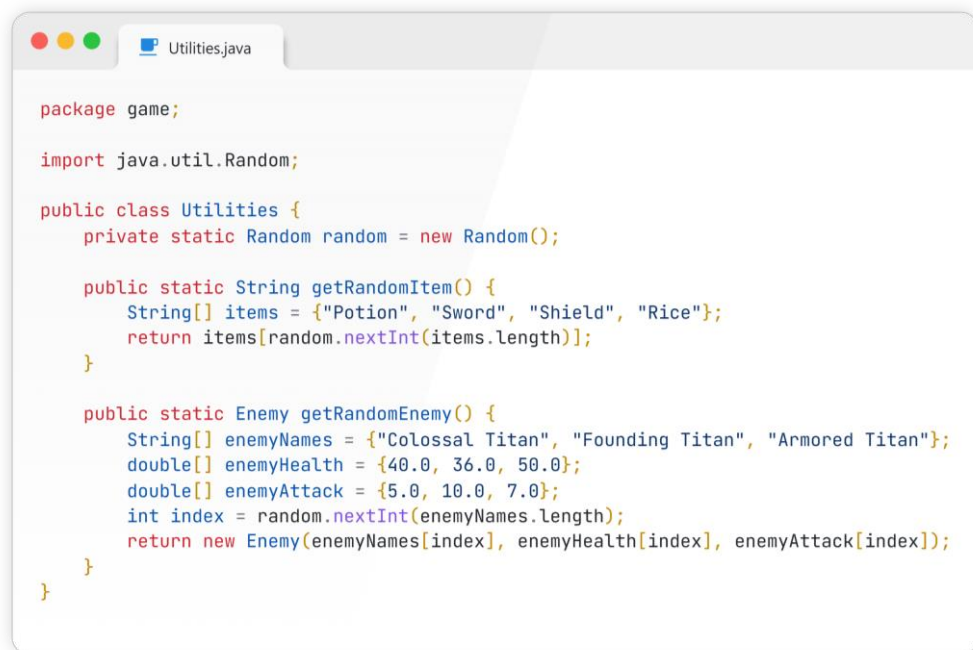
    public void reduceHealth(double damage) {
        this.health -= damage;
    }

    public double getAttackPower() {
        return attackPower;
    }
}
```

Metode `getName`, `getHealth`, dan `getAttackPower` digunakan untuk mengakses atribut-atribut tersebut. Metode `reduceHealth` mengurangi nilai kesehatan musuh berdasarkan jumlah kerusakan yang diterima.

2.5 Class Utilities

Kelas `Utilities` menyediakan utilitas acak untuk permainan. Kelas ini menggunakan objek `Random` untuk menghasilkan nilai acak. Metode `getRandomItem` mengembalikan item acak dari array item yang tersedia, seperti "Potion", "Sword", "Shield", atau "Rice".



```
package game;

import java.util.Random;

public class Utilities {
    private static Random random = new Random();

    public static String getRandomItem() {
        String[] items = {"Potion", "Sword", "Shield", "Rice"};
        return items[random.nextInt(items.length)];
    }

    public static Enemy getRandomEnemy() {
        String[] enemyNames = {"Colossal Titan", "Founding Titan", "Armored Titan"};
        double[] enemyHealth = {40.0, 36.0, 50.0};
        double[] enemyAttack = {5.0, 10.0, 7.0};
        int index = random.nextInt(enemyNames.length);
        return new Enemy(enemyNames[index], enemyHealth[index], enemyAttack[index]);
    }
}
```

Metode `getRandomEnemy` menghasilkan musuh acak dari daftar musuh yang telah ditentukan, seperti "Colossal Titan", "Founding Titan", atau "Armored Titan", dengan kesehatan dan kekuatan serangan yang sesuai. Setiap kali metode ini dipanggil, sebuah objek `Enemy` baru dibuat dengan atribut yang diambil secara acak dari array yang tersedia.

2. 6 Class Game

Kelas `Game` mengatur alur permainan dan interaksi antara pemain dan dunia permainan. Ini berisi atribut untuk pengguna saat ini, pemain, dan pemindai untuk input pengguna. Saat permainan dimulai, pengguna memilih peran sebagai "Magician" atau "Fighter" yang menentukan atribut awal pemain.

Metode `start` adalah titik masuk utama yang mengelola menu permainan dan memungkinkan pemain untuk membuka tas, memulai petualangan, menggunakan ramuan, atau keluar dari permainan. Setiap pilihan dalam menu memicu tindakan yang sesuai seperti menampilkan isi tas, memulai petualangan, menggunakan ramuan, atau mengakhiri permainan.

Metode `startAdventure` mengelola petualangan pemain, memberi mereka pilihan untuk bergerak maju, ke kanan, atau ke kiri. Saat pemain bergerak, ada peluang acak untuk menemukan musuh atau mendapatkan item. Jika musuh ditemukan, permainan berlanjut ke metode `battle`.

Metode `battle` menangani pertempuran antara pemain dan musuh. Pemain dapat memilih untuk menyerang atau melarikan diri. Jika menyerang, pemain menyebabkan kerusakan pada musuh, dan jika musuh masih hidup, musuh menyerang balik. Jika pemain mengalahkan musuh, mereka naik level dan atribut

mereka meningkat. Jika pemain memilih untuk melarikan diri, pertempuran berakhir.

```
Game.java

public void start() {
    System.out.println("Choose your role:");
    System.out.println("1. Magician");
    System.out.println("2. Fighter");
    System.out.print("Your choice: ");
    int roleChoice = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    String role = roleChoice == 1 ? "Magician" : "Fighter";
    player = new Player(role);
    boolean playing = true;

    while (playing) {
        System.out.println("1. Open backpack");
        System.out.println("2. Start Adventure");
        System.out.println("3. Use potion");
        System.out.println("4. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1:
                player.showBackpack();
                break;
            case 2:
                startAdventure();
                break;
            case 3:
                player.usePotion();
                break;
            case 4:
                playing = false;
                break;
        }
    }

    private void startAdventure() {
        boolean adventuring = true;
        while (adventuring) {
            System.out.println("1. Go straight");
            System.out.println("2. Move right");
            System.out.println("3. Move left");
            System.out.println("4. Exit Game");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            if (choice == 4) {
                adventuring = false;
                continue;
            }

            if (new java.util.Random().nextBoolean()) {
                Enemy enemy = Utilities.getRandomEnemy();
                System.out.println("Ups! Enemy founded");
                System.out.println("====Enemy Identity====");
                System.out.println("Enemy Name: " + enemy.getName());
                System.out.println("Enemy Blood: " + enemy.getHealth());
                System.out.println("Enemy Attack Power: " + enemy.getAttackPower());
                battle(enemy);
            } else {
                String item = Utilities.getRandomItem();
                System.out.println("You got an item: " + item);
                player.addItem(item);
            }
        }
    }

    private void battle(Enemy enemy) {
        while (enemy.getHealth() > 0 && player.getHealth() > 0) {
            System.out.println("Your health: " + player.getHealth());
            System.out.println("1. Attack");
            System.out.println("2. Run");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            if (choice == 2) {
                System.out.println("You ran away from the enemy");
                break;
            }

            double damage = player.getLevel() * 10;
            enemy.reduceHealth(damage);
            System.out.println("You hit the enemy for " + damage + " damage");

            if (enemy.getHealth() > 0) {
                player.reduceHealth(enemy.getAttackPower());
                System.out.println("The enemy hit you for " + enemy.getAttackPower() + " damage");
            } else {
                System.out.println("You defeated the enemy!");
                player.levelUp();
                System.out.println("Levels Up!");
                player.showStatus();
            }
        }
    }
}
```

Secara keseluruhan, kelas `Game` mengatur logika utama permainan, memungkinkan interaksi pengguna yang dinamis dan acak selama permainan.

2.7 Class Main



```
public static void main(String[] args) {
    users = User.loadUsers(); // Load existing users from file
    Scanner scanner = new Scanner(System.in);
    boolean running = true;

    while (running) {
        System.out.println("1. Login");
        System.out.println("2. Sign up");
        System.out.println("3. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1:
                login(scanner);
                break;
            case 2:
                signUp(scanner);
                break;
            case 3:
                running = false;
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
    scanner.close();
}

private static void login(Scanner scanner) {
    System.out.print("Enter username: ");
    String username = scanner.nextLine();
    System.out.print("Enter password: ");
    String password = scanner.nextLine();

    User user = users.get(username);
    if (user != null && user.getPassword().equals(password)) {
        System.out.println("Login successful!");
        Game game = new Game(user);
        game.start();
    } else {
        System.out.println("Invalid username or password.");
    }
}

private static void signUp(Scanner scanner) {
    System.out.print("Enter a username: ");
    String username = scanner.nextLine();
    System.out.print("Enter a password: ");
    String password = scanner.nextLine();

    if (users.containsKey(username)) {
        System.out.println("Username already taken. Please choose another.");
    } else {
        User newUser = new User(username, password);
        User.saveUser(newUser); // Save the new user to file
        users.put(username, newUser);
        System.out.println("User registered successfully!");
    }
}
```

Kelas 'Main' mengatur alur utama aplikasi, menyediakan antarmuka pengguna untuk login, pendaftaran, dan keluar. Pada awalnya, data pengguna dimuat dari file menggunakan metode 'User.loadUsers()'. Dengan menggunakan objek

`Scanner`, program ini menampilkan menu utama yang memungkinkan pengguna untuk login, mendaftar, atau keluar.

Dalam metode `main`, sebuah loop berjalan selama `running` bernilai `true`, menampilkan opsi menu dan memproses input pengguna. Jika pengguna memilih untuk login, metode `login` dipanggil. Di sini, pengguna diminta untuk memasukkan username dan password. Jika kredensial valid, pengguna masuk ke permainan dengan menciptakan objek `Game` dan memulai permainan.

Metode `signUp` mengurus pendaftaran pengguna baru. Setelah memasukkan username dan password, program memeriksa apakah username sudah ada dalam sistem. Jika belum, pengguna baru dibuat, disimpan ke file, dan ditambahkan ke peta pengguna.

Program ini menangani interaksi pengguna dengan cara yang mudah dimengerti, memungkinkan pengguna untuk mendaftar dan login, dan memulai permainan setelah login berhasil.

2. 8 Launch program

1. Registrasi

```
1. Login
2. Sign up
3. Exit
Enter your choice: 2
Enter a username: user3
Enter a password: 321
User registered successfully!
```

Sebelum memulai permainan, user diminta untuk mendaftar terlebih dahulu jika belum membuat akun dengan mengisi username dan password. Namun jika sudah memiliki akun, user bisa langsung login dengan akun yang telah didaftarkan.

2. Login

```
1. Login
2. Sign up
3. Exit
Enter your choice: 1
Enter username: user3
Enter password: 321
Login successful!
```

User diminta login dengan akun yang telah didaftarkannya. Setelah berhasil login maka user akan diminta untuk memilih role nya.

```
Choose your role:  
1. Magician  
2. Fighter  
Your choice: 2
```

Setelah role nya telah dipilih maka user sudah bisa menikmati permainannya.

3. Bermain

- Membuka tas

```
1. Open backpack  
2. Start Adventure  
3. Use potion  
4. Exit  
Enter your choice: 1  
Items in Backpack: []
```

- Memulai Petualangan

```
1. Open backpack  
2. Start Adventure  
3. Use potion  
4. Exit  
Enter your choice: 2  
1. Go straight  
2. Move right  
3. Move left  
4. Exit Game  
Enter your choice: 2  
You got an item: Sword  
1. Go straight  
2. Move right  
3. Move left  
4. Exit Game  
Enter your choice: 3  
You got an item: Sword
```


- Menemukan Musuh pertama

```
1. Go straight
2. Move right
3. Move left
4. Exit Game
Enter your choice: 1
Upss! Enemy founded
===Enemy Identity=====
Enemy Name: Founding Titan
Enemy Blood: 36.0
Enemy Attack Power: 10.0
Your health: 150.0
```

- Menyerang Musuh

```
1. Attack
2. Run
Enter your choice: 1
You hit the enemy for 10.0 damage
The enemy hit you for 10.0 damage
Your health: 140.0
1. Attack
2. Run
Enter your choice: 1
You hit the enemy for 10.0 damage
The enemy hit you for 10.0 damage
Your health: 130.0
1. Attack
2. Run
Enter your choice: 1
You hit the enemy for 10.0 damage
The enemy hit you for 10.0 damage
Your health: 120.0
1. Attack
2. Run
Enter your choice: 1
You hit the enemy for 10.0 damage
You defeated the enemy!
Levels Up!
Your health: 170.0
Your mana: 70.0
Your level: 2
```

- Membuka tas

```
1. Open backpack
2. Start Adventure
3. Use potion
4. Exit
Enter your choice: 1
Items in Backpack: [Sword, Sword]
```

- Exit game

```
1. Open backpack
2. Start Adventure
3. Use potion
4. Exit
Enter your choice: 4
1. Login
2. Sign up
3. Exit
Enter your choice: 3
PS C:\Users\benon\OneDrive\Documents\
```

- Melihat database.txt

```
database.txt
1 user1,123
2 user2,123
3 user3,321
4
```

BAB III

KESIMPULAN

Kesimpulan dari game petualangan berbasis teks ini adalah bahwa game ini menggabungkan berbagai konsep dasar pemrograman berorientasi objek (OOP) dalam Java, seperti inheritance, encapsulation, dan penggunaan koleksi serta I/O file. Game ini menyediakan antarmuka pengguna sederhana yang memungkinkan pemain untuk mendaftar dan login, memilih peran mereka, dan kemudian menjelajahi dunia game dengan menghadapi berbagai musuh dan mengumpulkan item.

Fitur utama dari game ini meliputi:

- Manajemen Pengguna: Pengguna dapat mendaftar dan login dengan username dan password. Data pengguna disimpan dan dimuat dari file untuk memastikan persistensi.
- Pemilihan Peran: Pemain dapat memilih antara dua peran, Magician atau Fighter, dengan atribut kesehatan dan mana yang berbeda.
- Petualangan dan Pertarungan: Pemain dapat menjelajah dunia game, menemukan item acak, atau bertemu dengan musuh. Dalam pertarungan, pemain dapat memilih untuk menyerang atau melarikan diri.
- Peningkatan Level: Pemain bisa meningkatkan level mereka setelah mengalahkan musuh, yang meningkatkan atribut kesehatan dan mana mereka.

Kelebihan dari game ini adalah penerapan berbagai konsep OOP yang baik, struktur kode yang terorganisir, dan penggunaan I/O file untuk manajemen pengguna. Namun, game ini masih dapat ditingkatkan dengan menambahkan fitur-fitur seperti lebih banyak jenis musuh, lebih banyak item, dan mekanisme permainan yang lebih kompleks untuk meningkatkan pengalaman bermain.