

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 5**



**NAMA : M TAQIYYUDDIN F
NIM : 105219039**

**PROGRAM STUDI ILMU KOMPUTER
FAKULTAS SAINS DAN ILMU KOMPUTER
UNIVERSITAS PERTAMINA 2021**

1. Pendahuluan

a. Inti Sari

Pada praktikum di modul ini, dibuat suatu program yang membuat suatu class abstract dan interface. Class abstrak adalah suatu class yang masih dalam bentuk abstrak, dimana terdapat method yang belum memiliki suatu perintah tertentu. Sedangkan class interface adalah suatu class yang menghubungkan proses suatu class dengan perintah yang menjalankannya. Pada percobaan ini dibuat class abstract berupa Character yang memiliki sub-class Hero, dan class interface Animation yang memiliki sub-class HeroAnimation.

b. Rumusan Masalah

- Bagaimana class abstract bekerja ?
- Bagaimana class interface bekerja ?

2. Pembahasan

```
public abstract class Character {  
    protected double healthBase = 1.5;  
    protected double damageBase = 1.2;  
    protected String nama;
```

Pada percobaan ini, dideklarasikan suatu attribute pada class abstract Character untuk menentukan nyawa dan damage base dari character tersebut (character ini bisa menjadi npc, hero, dst) beserta namanya.

```
protected void setNama(String nama) {  
    this.nama = nama;  
}  
  
protected String getNama() {  
    return this.nama;  
}
```

Pada percobaan ini, dibuat suatu method setter dan getter dari nama tersebut.

```
abstract void animation(Animation animasi);  
abstract void specialAction(); |
```

Pada percobaan ini, dibuat method abstract animation yang menerima jenis animation karakter tersebut dapat lakukan, beserta special action yang unik untuk setiap class yang menggunakan abstrak ini berdasarkan classnya

```
public class Hero extends Character {  
    private int health;  
    private int damage;  
    private Animation animasi;
```

Pada percobaan ini, dideklarasikan atribut dari class hero yang menggunakan abstract karakter tersebut, terdiri dari health dan damage sebagai nyawa dan damage dari hero tersebut. Beserta atribut animasi yang menyimpan class animasi yang dapat dijalankan.

```
public Hero(String nama, int health, int damage) {  
    setNama(nama);  
    this.health = (int)(healthBase * health);  
    this.damage = (int)(damageBase * damage);  
}
```

Pada percobaan ini, dibuat suatu constructor yang memberikan nama dari hero tersebut, beserta memberikan nilai nyawa dan damage dari hero dimana nilai yang diberikan saat membentuk objek tersebut dikali dengan nyawa dan damage berturut-turut, lalu dibulatkan.

```
public void menyerang() {  
    System.out.println("Memberikan damage sebesar : " + this.damage);  
    specialAction();  
    animasi.fight();  
    System.out.println();  
}
```

Pada percobaan ini, dibuat suatu method yang menerima tindakan hero tersebut untuk menyerang, dengan menampilkan damage yang dia berikan beserta tindakan special yang dapat dia lakukan. Serangan tersebut akan menjalankan method fight pada objek animasi dari class animation tersebut.

```
public Animation animasi() {  
    return this.animasi;  
}  
  
@Override  
public void animation(Animation animasi) {  
    this.animasi = animasi;  
}
```

Pada percobaan ini, dibuat proses method abstract tersebut yaitu memberikan nilai objek animasi di class tersebut dengan argument yang akan diberikan, serta method animasi yang memberikan objek animasi yang telah didapatkan.

```

@Override
public void specialAction() {
    double random = Math.random();
    //System.out.println(random);
    if(random < 0.2) {
        System.out.println("Mendapatkan ekstra damage sebesar : " + (this.damage * 0.25));
    }
}

```

Pada percobaan ini, dibuat kerangka proses method abstract yang dimiliki oleh class hero, dimana saat mereka menyerang, ada kemungkinan mereka memberikan damage lebih, dengan syarat jika nilai random yang diperoleh kurang dari 0.2 (dimana math.random() memberikan nilai antara 0.0 sampai 1.0).

```

public interface Animation {
    void idle();
    void rest();
    void fight();
}

```

Pada percobaan ini, dibuat suatu class animation dimana terdiri dari tiga jenis tindakan, yaitu mode tenang / idle, istirahat maupun fight yang akan menampilkan tindakan tersebut berdasarkan class yang mengimplementasikannya.

```

public class HeroAnimation implements Animation {
    private boolean isRest;

    @Override
    public void idle() {
        isRest = false;
        System.out.println("Hero sedang mendalami skill\n");
    }

    @Override
    public void rest() {
        isRest = true;
        System.out.println("Hero sedang beristirahat\n");
    }

    @Override
    public void fight() {
        if(!isRest) {
            System.out.println("Hero menyerang!!\n");
        } else {
            System.out.println("Hero sedang beristirahat, tidak bisa menyerang\n");
        }
    }
}

```

Pada percobaan ini, dibuat suatu class yang mengimplementasikan class animation untuk hero bernama HeroAnimation. Class ini memiliki atribut yaitu isRest yang mengecek kondisi apakah hero tersebut sedang beristirahat atau tidak.

Pada saat hero sedang idle, maka isRest jadi false dan menampilkan tindakan restnya, begitu pula sebaliknya pada saat dijalankan method rest. Pada saat hero menjalankan method fight, akan dicek jika hero tersebut tidak sedang beristirahat maka hero dapat menyerang, jika hero sedang beristirahat, maka damage yang telah dihitung tidak akan dieksekusi karena hero tersebut sedang beristirahat.

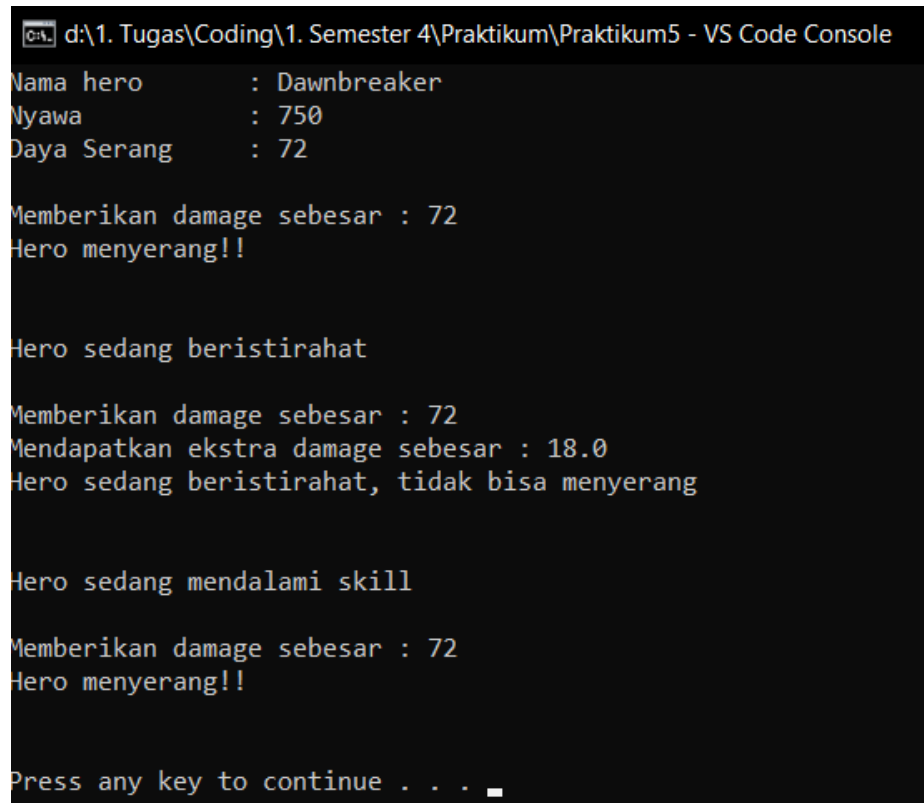
```
public static void main(String[] args) {  
    Hero dawnbreaker = new Hero("Dawnbreaker", 500, 60);  
    HeroAnimation heroAnimasi = new HeroAnimation();  
  
    dawnbreaker.animation(heroAnimasi);  
  
    dawnbreaker.stats();  
    dawnbreaker.menyerang();  
    dawnbreaker.animasi().rest();  
    dawnbreaker.menyerang();  
    dawnbreaker.animasi().idle();  
    dawnbreaker.menyerang();  
}
```

Pada percobaan ini, dibuat suatu objek hero bernama dawnbreaker, dengan parameter nama dawnbreaker, dengan darah dan damage berturut-turut 500 dan 60. Beserta membuat suatu objek animasi hero. Lalu, objek dawnbreaker menerima objek animasi hero yang mengimplementasi class animasi untuk dijalankan tindakan animasinya.

3. Kesimpulan

- Class abstract digunakan pada saat pembuatan suatu class yang memerlukan kebutuhan yang tidak tentu pada setiap class ataupun suatu method yang tidak kita ketahui secara pasti kegunaanya, seperti pada class character, dimana specialAction dan Animation tersebut berdasarkan kegunaan animasi hero, jika semisal dibuat suatu class NPC, maka specialAction dan Animation dapat berbeda.
- Class interface digunakan pada saat menampilkan tindakan pada suatu class, dimana tindakan tersebut proses eksekusinya berbeda-beda, namun memiliki fitur tindakan / method yang sama.

4. Tes Akhir

A screenshot of a VS Code console window with a black background and white text. The title bar at the top reads "d:\1. Tugas\Coding\1. Semester 4\Praktikum\Praktikum5 - VS Code Console". The output text is as follows:

```
Nama hero      : Dawnbreaker  
Nyawa         : 750  
Daya Serang   : 72  
  
Memberikan damage sebesar : 72  
Hero menyerang!!  
  
Hero sedang beristirahat  
  
Memberikan damage sebesar : 72  
Mendapatkan ekstra damage sebesar : 18.0  
Hero sedang beristirahat, tidak bisa menyerang  
  
Hero sedang mendalami skill  
  
Memberikan damage sebesar : 72  
Hero menyerang!!  
  
Press any key to continue . . . _
```

Gambar 4.1. Hasil Output Dari Program Tersebut

Folder package terdapat pada file terkompresi.