

UAS
PEMROGRAMAN BERORIENTASI
OBJEK

DOSEN PENGAMPU
SAYEKTI HARITS SURYAWAN, S.Kom., M.Kom



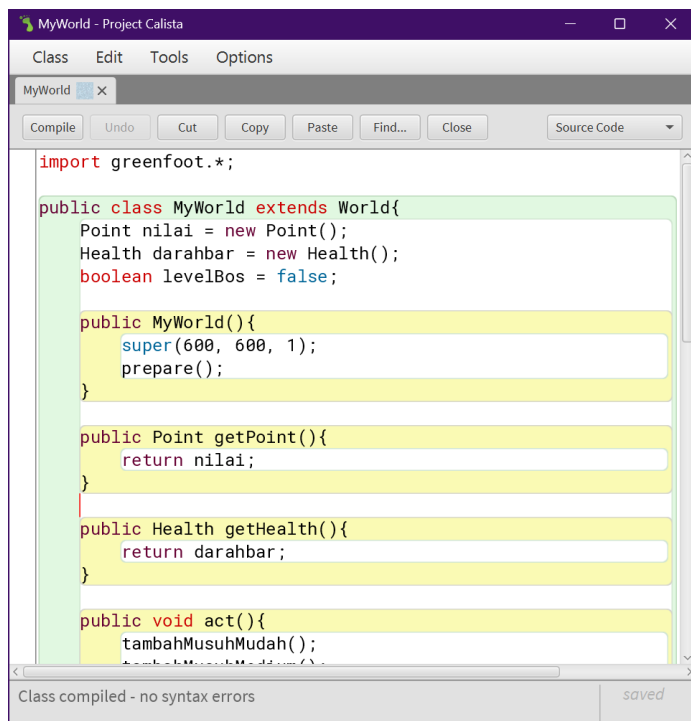
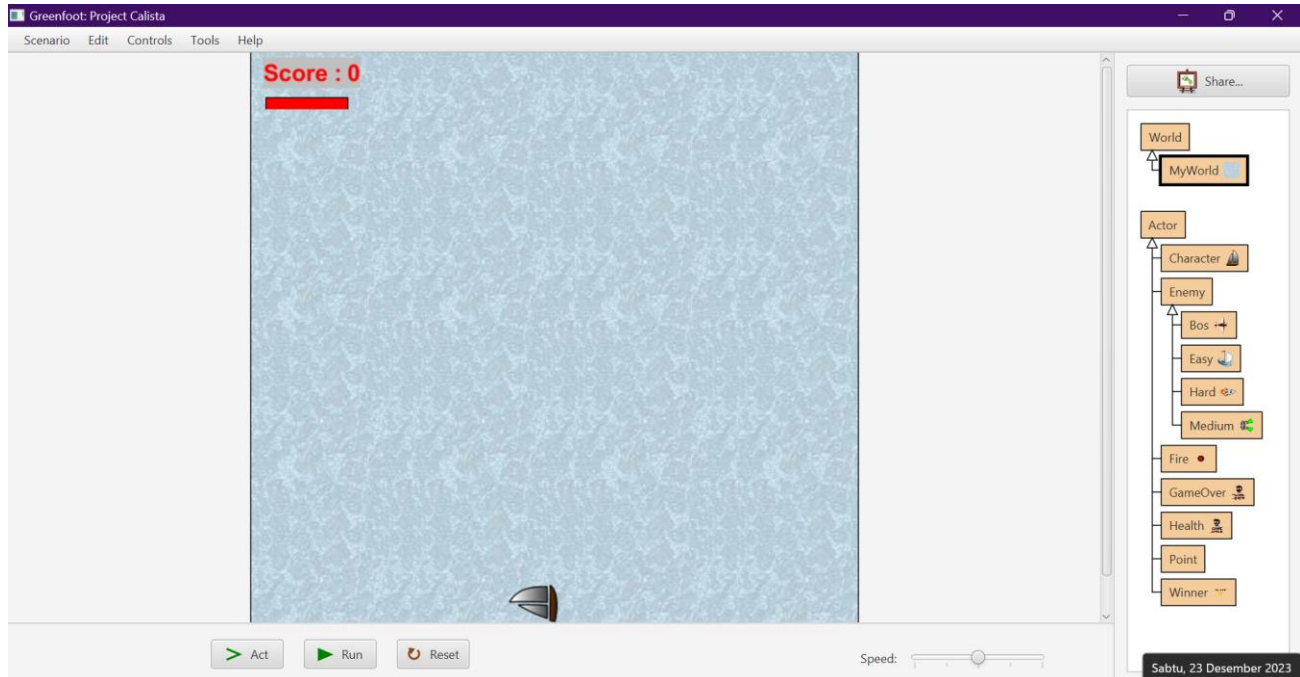
DISUSUN OLEH :

- **MUHAMMAD AWANG OSAMA DARMAWAN – 2211102441010**
- **CALISTA AMANDA SYACHMIRAL - 2211102441008**
- **MUHAMMAD LUTHFI SOFIAN - 2211102441062**
- **MUHAMMAD ANDHIKA JULIAN – 2211102441107**

S1 TEKNIK INFORMATIKA

FAKULTAS SAINTS DAN TEKNOLOGI
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR
2023

Tampilan game:



Deklarasi Variabel :

Jadi kami membuat Variabel nilai dan darahbar digunakan yang dimana berfungsi untuk menyimpan objek Point dan Health. Variabel levelBos digunakan untuk menandai apakah level bos telah dicapai.

Konstruktor MyWorld :

Konstruktor inisialisasi dunia permainan dengan ukuran 600x600 piksel dan layer 1, kemudian memanggil metode prepare() untuk menyiapkan elemen-elemen awal.

Getter untuk Point dan Health :

Metode-metode ini digunakan untuk mendapatkan referensi ke objek Point dan Health dari kelas lain.

Metode act :

Metode ini dipanggil setiap frame dan bertanggung jawab untuk menambahkan musuh-musuh dengan tingkat kesulitan yang berbeda serta memeriksa apakah sudah waktunya untuk menambahkan bos.

Metode tambahMusuhMudah, tambahMusuhMedium, tambahMusuhSusah :

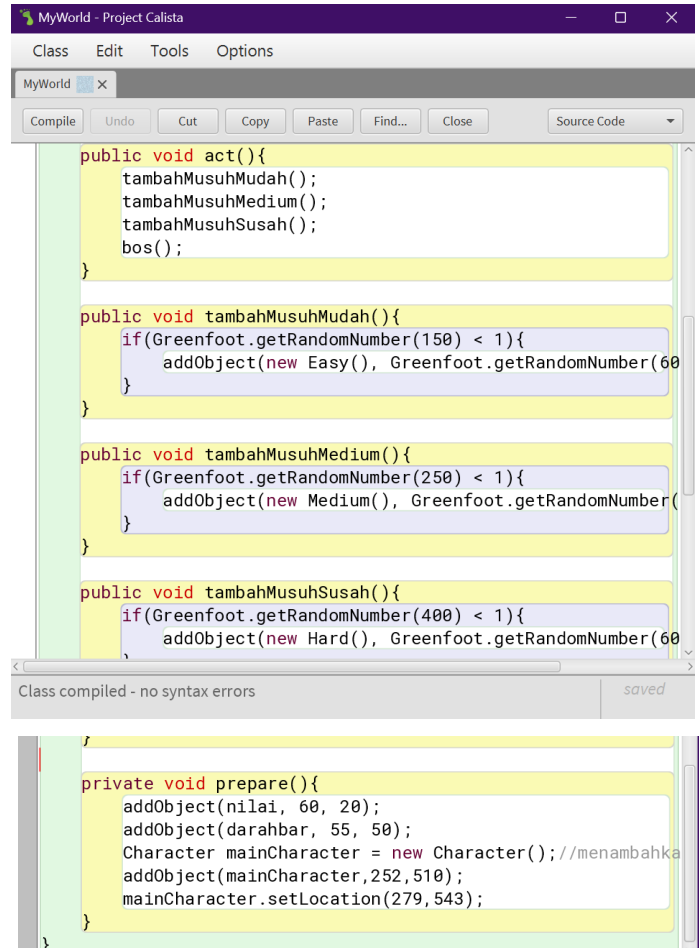
Metode-metode ini menambahkan musuh dengan tingkat kesulitan yang berbeda berdasarkan angka acak.

Metode bos :

Metode ini menambahkan bos ke dunia permainan jika pemain mencapai poin tertentu.

Metode prepare :

Metode ini menyiapkan elemen-elemen awal dalam permainan, termasuk objek Point, objek Health, dan karakter utama (mainCharacter).



```
public void act(){
    tambahMusuhMudah();
    tambahMusuhMedium();
    tambahMusuhSusah();
    bos();
}

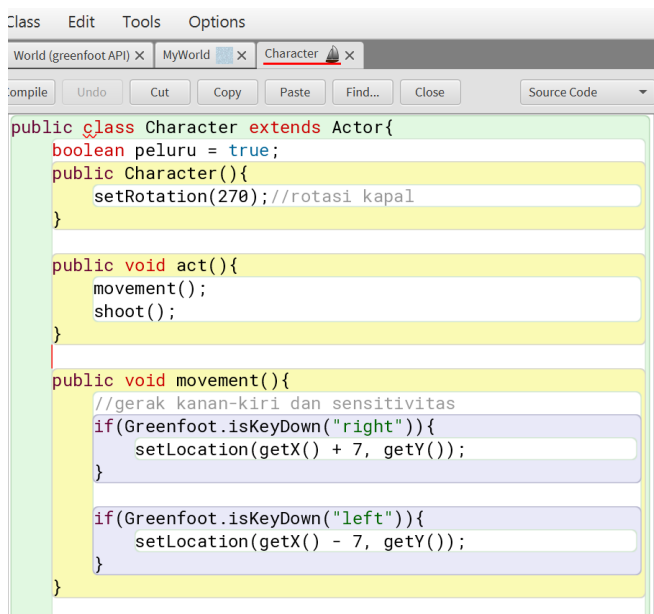
public void tambahMusuhMudah(){
    if(Greenfoot.getRandomNumber(150) < 1){
        addObject(new Easy(), Greenfoot.getRandomNumber(600));
    }
}

public void tambahMusuhMedium(){
    if(Greenfoot.getRandomNumber(250) < 1){
        addObject(new Medium(), Greenfoot.getRandomNumber(600));
    }
}

public void tambahMusuhSusah(){
    if(Greenfoot.getRandomNumber(400) < 1){
        addObject(new Hard(), Greenfoot.getRandomNumber(600));
    }
}

private void prepare(){
    addObject(nilai, 60, 20);
    addObject(darahbar, 55, 50);
    Character mainCharacter = new Character();//menambahkan karakter
    addObject(mainCharacter, 252, 510);
    mainCharacter.setLocation(279, 543);
}
```

Kemudian pada Aktor kami menambahkan class Character.



```
public class Character extends Actor{
    boolean peluru = true;
    public Character(){
        setRotation(270);//rotasi kapal
    }

    public void act(){
        movement();
        shoot();
    }

    public void movement(){
        //gerak kanan-kiri dan sensitivitas
        if(Greenfoot.isKeyDown("right")){
            setLocation(getX() + 7, getY());
        }

        if(Greenfoot.isKeyDown("left")){
            setLocation(getX() - 7, getY());
        }
    }
}
```

Deklarasi Kelas Character :

Mendefinisikan kelas Character yang merupakan turunan dari kelas Actor dalam Greenfoot.

Variabel boolean peluru :

Variabel ini digunakan untuk memastikan bahwa karakter hanya dapat menembak setelah tembakan sebelumnya selesai.

Konstruktur Character :

Konstruktur mengatur rotasi karakter menjadi 270, mungkin untuk menunjukkan bahwa karakter menghadap ke atas.

Metode act :

Metode ini dipanggil setiap frame dan memanggil metode movement() untuk mengatur gerakan karakter dan shoot() untuk menangani tembakan.

Metode movement :

Metode ini mengatur gerakan karakter ke kanan dan kiri berdasarkan input keyboard. Sensitivitas gerakan ditentukan oleh nilai 7.

Metode Signature : “ Public void shoot() ”

Ini adalah sebuah metode yang tidak mengembalikan nilai (void) dan bernama “shoot” . Metode ini melakukan fungsi untuk menembakkan suatu objek atau melakukan aksi yang terkait dengan menembak.

Kemudian pada Enemy kami menambahkan Class Enemy yang merupakan turunan dari kelas aktor.

```
public class Enemy extends Actor{  
    public void act(){  
    }  
  
    public void movementMusuhMudah(){  
        //kecepatan musuh + 2  
        setLocation(getX(), getY() + 2);  
    }  
  
    public void movementMusuhSedang(){  
        //kecepatan musuh + 4  
        setLocation(getX(), getY() + 4);  
    }  
  
    public void movementMusuhSusah(){  
        //kecepatan musuh + 3  
        setLocation(getX(), getY() + 3);  
    }  
  
    public void movementBos(){  
        //kecepatan musuh + 5  
        setLocation(getX(), getY() + 5);  
    }  
}
```

Metode Act () :

Metode ini adalah metode yang dipanggil secara berkala oleh lingkungan permainan (framework) untuk mengatur perilaku objek musuh (enemy) dalam permainan.

Metode movementMusuhMudah () :

Metode ini mengatur pergerakan musuh dengan Tingkat kesulitan yang rendah. Dalam implementasinya, posisi musuh diubah dengan cara menggesernya kebawah (menambah nilai ‘ getY()’ sebesar 2)

Metode movementMusuhSedang () :

Metode ini mengatur pergerakan musuh dengan Tingkat sedang. Seperti sebelumnya, perubahan posisi musuh juga dilakukan dengan menggesernya kebawah (menambah nilai ‘ getY()’ sebesar 4).

Metode movementMusuhSusah ():

Metode ini mengatur pergerakan musuh dengan Tingkat kesulitan yang tinggi. Pada dasarnya, pergerakan ini mirip dengan Tingkat kesulitan sedang, hanya saja perubahan posisi musuh dilakukan dengan menambah nilai ‘ getY()’ sebesar 3.

Metode movementBos():

Metode ini mengatur pergerakan bos musuh dengan Tingkat kesulitan yang lebih tinggi dari sebelumnya. Perubahan psoso musuh dilakukan dengan menambah nilai ‘ getY()’ sebesar 5.

```

public void musuhHilangEasy(){
    //jika musuh melewati getY = 599 maka hilang
    if(getY() == 599 ){
        World beranda = getWorld();
        MyWorld utama = (MyWorld)beranda;
        Health darahbar = utama.getHealth();
        darahbar.loseBlood();
        getWorld().removeObject(this);
    }
}

public void musuhHilangMedium(){
    //jika musuh melewati getY = 599 maka hilang
    if(getY() == 599 ){
        World beranda = getWorld();
        MyWorld utama = (MyWorld)beranda;
        Health darahbar = utama.getHealth();
        darahbar.loseBlood();
        darahbar.loseBlood();
        darahbar.loseBlood();
        getWorld().removeObject(this);
    }
}

```

Metode musuhHilangEasy() :

Metode ini menentukan kondisi Ketika musuh dengan Tingkat kesulitan yang mudah “Hilang” dari dunia permainan. Jika posisi Y musuh sama dengan 599 (batas bawah tertentu dalam permainan), maka aksi ini akan dijalankan :

- Objek ‘World’ bernama ‘beranda’ diambil dari dunia permainan.
- ‘beranda’ dikonversi menjadi objek ‘MyWorld’.
- Objek ‘Health’ dengan nama ‘darahbar’ diambil dari ‘MyWorld’
- loseBlood() dipanggil pada darahbar untuk mengurangi "darah" atau kesehatan dalam permainan.
- Objek this (musuh) dihapus dari dunia permainan menggunakan removeObject(this).

Metode musuhHilangMedium() :

- Metode ini menentukan kondisi ketika musuh dengan tingkat kesulitan medium "hilang" dari dunia permainan.
- Jika posisi Y musuh sama dengan 599 (mungkin batas bawah tertentu dalam permainan), maka aksi ini akan dijalankan:
- Objek World bernama beranda diambil dari dunia permainan.
- beranda dikonversi menjadi objek MyWorld.
- Objek Health dengan nama darahbar diambil dari MyWorld.
- loseBlood() dipanggil tiga kali pada darahbar untuk mengurangi "darah" atau kesehatan dalam permainan.
- Objek this (musuh) dihapus dari dunia permainan.


```

public class Bos extends Enemy{
    int headshot = 5;

    public Bos(){
        setRotation(90);
    }

    public void act(){
        movementBos();
        hit();
    }

    public void hit(){
        //menambahkan objek musuhHard, jika musuhMedium tidak sama dengan null (Hit 2x),
        //maka musuhHard menghilang (Jika tertembak 2x = Hilang)
        Actor musuhBos = getOneIntersectingObject(Fire.class);
        if(musuhBos != null){
            getWorld().removeObject(musuhBos);
            hasil();
            headshot--;
        }
        //hit 5x
        if(headshot == 0){
            getWorld().removeObject(this);
        }
        else{
            bosHilang();
        }
    }
}

```

Variabel headshot:

- Variabel ini bertipe integer dan diinisialisasi dengan nilai 5.
- variabel ini digunakan untuk melacak jumlah "headshot" atau tembakan ke bagian kepala yang diperlukan agar bos ini dihilangkan dari permainan.

Konstruktor Bos():

- Saat objek Bos dibuat, konstruktor ini dipanggil.
- Konstruktor ini mengatur rotasi bos ke 90 derajat, untuk menentukan arah awal pergerakan atau orientasi dari bos.

Metode act():

- Metode ini dipanggil secara berkala oleh lingkungan permainan (framework) untuk mengatur perilaku objek bos dalam permainan.
- Di dalam metode act(), terdapat pemanggilan metode movementBos() yang merupakan bagian dari kelas Enemy, dan juga pemanggilan metode hit() yang tampaknya merupakan logika yang terkait dengan terkena tembakan.

Metode hit():

- Metode ini memeriksa apakah objek bos terkena tembakan.
- Jika ada objek Fire (mungkin peluru) yang menyentuh objek bos (getOneIntersectingObject(Fire.class)), maka objek peluru dihapus dari dunia permainan dengan getWorld().removeObject(musuhBos).
- Selanjutnya, metode hasil() dipanggil, yang kemungkinan bertanggung jawab untuk menambah skor atau nilai dalam permainan, dan nilai variabel headshot dikurangi satu.

Logika penghilangan bos:

- Jika headshot mencapai nilai 0 (artinya bos telah terkena tembakan sebanyak 5 kali), maka objek bos dihapus dari dunia permainan dengan getWorld().removeObject(this).
- Jika headshot belum mencapai 0, metode bosHilang() dipanggil, yang berhubungan dengan logika terkait ketika bos harus dihapus dalam situasi tertentu dalam permainan.

```

import greenfoot.*;
//musuh Easy bernilai 1 poin dan hanya 1 hit
public class Easy extends Enemy{
    public Easy(){
        setRotation(90);
    }

    public void act(){
        movementMusuhMudah();
        hitEasy();
    }

    public void hitEasy(){
        //menambahkan objek musuhEasy, jika musuhEasy tidak sama dengan null,
        //maka musuhEasy menghilang (Jika tertembak = Hilang)
        Actor musuhEasy = getOneIntersectingObject(Fire.class);
        if(musuhEasy != null){
            getWorld().removeObject(musuhEasy);
            hasil();
            getWorld().removeObject(this);
        }
        //jika musuhEasy tidak tertembak sama dengan null,
        //maka akan menjalankan perintah musuhHilang
        else{
            musuhHilangEasy();
        }
    }
}

```

Konstruktor Easy():

- Ketika objek Easy dibuat, konstruktor ini dipanggil.
- Konstruktor ini mengatur rotasi musuh dengan tingkat kesulitan mudah ke 90 derajat, untuk menentukan arah awal pergerakan atau orientasi dari musuh tersebut.

Metode act():

- Metode ini dipanggil secara berkala oleh lingkungan permainan (framework) untuk mengatur perilaku objek musuh dalam permainan.
- Di dalam metode act(), terdapat pemanggilan metode movementMusuhMudah() yang merupakan bagian dari kelas Enemy, dan juga pemanggilan metode hitEasy() yang tampaknya merupakan logika yang terkait dengan deteksi terkena tembakan dan penghapusan musuh.

Metode hitEasy():

- Metode ini memeriksa apakah objek musuh terkena tembakan atau tidak.
- Jika ada objek Fire (mungkin peluru) yang menyentuh objek musuh (getOneIntersectingObject(Fire.class)), maka objek peluru dihapus dari dunia permainan dengan getWorld().removeObject(musuhEasy).
- Kemudian, metode hasil() dipanggil, yang mungkin bertanggung jawab untuk menambah skor atau nilai dalam permainan.
- Setelah itu, objek musuh dengan tingkat kesulitan mudah juga dihapus dari dunia permainan menggunakan getWorld().removeObject(this).

Logika ketika musuh tidak terkena tembakan:

- Jika tidak ada objek Fire yang menyentuh musuh, maka akan dijalankan perintah musuhHilangEasy(), yang kemungkinan berhubungan dengan logika ketika musuh harus dihapus dari permainan dalam situasi tertentu.


```

public class Hard extends Enemy{
    int tigaHit = 3;

    public Hard(){
        setRotation(90);
    }

    public void act(){
        movementMusuhSusah();
        hit();
    }

    public void hit(){
        //menambahkan objek musuhHard, jika musuhMedium tidak sama dengan null (Hit 2x),
        //maka musuhHard menghilang (Jika tertembak 2x = Hilang)
        Actor musuhHard = getOneIntersectingObject(Fire.class);
        if(musuhHard != null){
            getWorld().removeObject(musuhHard);
            hasil();
            tigaHit--;
        }
        //hit 3x
        if(tigaHit == 0){
            getWorld().removeObject(this);
        }
        else{
            musuhHilangHard();
        }
    }
}

```

Variabel tigaHit:

- Variabel ini bertipe integer dan diinisialisasi dengan nilai 3.
- Kemungkinan digunakan untuk melacak jumlah hit atau tembakan yang diperlukan agar musuh ini dihilangkan dari permainan.

Konstruktor Hard():

- Saat objek Hard dibuat, konstruktor ini dipanggil.
- Konstruktor ini mengatur rotasi musuh dengan tingkat kesulitan tinggi ke 90 derajat, mungkin untuk menentukan arah awal pergerakan atau orientasi dari musuh tersebut.

Metode act():

- Metode ini dipanggil secara berkala oleh lingkungan permainan (framework) untuk mengatur perilaku objek musuh dalam permainan.
- Di dalam metode act(), terdapat pemanggilan metode movementMusuhSusah() yang merupakan bagian dari kelas Enemy, dan juga pemanggilan metode hit() yang tampaknya merupakan logika yang terkait dengan deteksi terkena tembakan dan penghapusan musuh.

Metode hit():

- Metode ini memeriksa apakah objek musuh terkena tembakan atau tidak.
- Jika ada objek Fire (mungkin peluru) yang menyentuh objek musuh (getOneIntersectingObject(Fire.class)), maka objek peluru dihapus dari dunia permainan dengan getWorld().removeObject(musuhHard).
- Setelah itu, metode hasil() dipanggil, yang mungkin bertanggung jawab untuk menambah skor atau nilai dalam permainan.
- Variabel tigaHit dikurangi satu.
- Logika penghilangan musuh tingkat kesulitan tinggi:
- Jika tigaHit mencapai nilai 0 (artinya musuh telah terkena tembakan sebanyak 3 kali), maka objek musuh dengan tingkat kesulitan tinggi dihapus dari dunia permainan menggunakan getWorld().removeObject(this).
- Jika tigaHit belum mencapai 0, maka dipanggil metode musuhHilangHard(), yang kemungkinan berhubungan dengan logika penghapusan musuh dalam situasi tertentu dalam permainan.

```

public class Medium extends Enemy{
    int duaHit = 2;

    public Medium(){
        setRotation(90);
    }

    public void act(){
        movementMusuhSedang();
        hit();
    }

    public void hit(){
        //menambahkan objek musuhMedium, jika musuhMedium tidak sama dengan null (Hit 2x),
        //maka musuhMedium menghilang (Jika tertembak 2x = Hilang)
        Actor musuhMedium = getOneIntersectingObject(Fire.class);
        if(musuhMedium != null){
            getWorld().removeObject(musuhMedium);
            hasil();
            duaHit--;
        }
        //hit 2x
        if(duaHit == 0){
            getWorld().removeObject(this);
        }
        else{
            musuhHilangMedium();
        }
    }
}

```

Variabel duaHit:

- Variabel ini bertipe integer dan diinisialisasi dengan nilai 2.
- Kemungkinan digunakan untuk melacak jumlah hit atau tembakan yang diperlukan agar musuh ini dihilangkan dari permainan.

Konstruktur Medium():

- Saat objek Medium dibuat, konstruktur ini dipanggil.
- Konstruktur ini mengatur rotasi musuh dengan tingkat kesulitan sedang ke 90 derajat, mungkin untuk menentukan arah awal pergerakan atau orientasi dari musuh tersebut.

Metode act():

- Metode ini dipanggil secara berkala oleh lingkungan permainan (framework) untuk mengatur perilaku objek musuh dalam permainan.
- Di dalam metode act(), terdapat pemanggilan metode movementMusuhSedang() yang merupakan bagian dari kelas Enemy, dan juga pemanggilan metode hit() yang tampaknya merupakan logika yang terkait dengan deteksi terkena tembakan dan penghapusan musuh.

Metode hit():

- Metode ini memeriksa apakah objek musuh terkena tembakan atau tidak.
- Jika ada objek Fire (mungkin peluru) yang menyentuh objek musuh (getOneIntersectingObject(Fire.class)), maka objek peluru dihapus dari dunia permainan dengan getWorld().removeObject(musuhMedium).
- Setelah itu, metode hasil() dipanggil, yang mungkin bertanggung jawab untuk menambah skor atau nilai dalam permainan.
- Variabel duaHit dikurangi satu.

Logika penghilangan musuh tingkat kesulitan sedang:

- Jika duaHit mencapai nilai 0 (artinya musuh telah terkena tembakan sebanyak 2 kali), maka objek musuh dengan tingkat kesulitan sedang dihapus dari dunia permainan menggunakan getWorld().removeObject(this).
- Jika duaHit belum mencapai 0, maka dipanggil metode musuhHilangMedium(), yang kemungkinan berhubungan dengan logika penghapusan musuh dalam situasi tertentu dalam permainan.

```
import greenfoot.*;

public class Fire extends Actor{
    public void act(){
        moveShoot();
        disappear();
    }

    public void moveShoot(){
        //arah tembak
        setLocation(getX(), getY() - 5);
    }

    public void disappear(){
        //peluru hilang jika melewati batas
        if(getY() == 0 ){
            getWorld().removeObject(this);
        }
    }
}
```

Metode act():

- Metode act() adalah metode yang dipanggil secara berkala oleh lingkungan permainan (framework) untuk mengatur perilaku objek Fire.
- Di dalam metode act(), terdapat pemanggilan dua metode lain: moveShoot() dan disappear().

Metode moveShoot():

- Metode ini mengatur pergerakan peluru dengan mengubah posisinya ke atas, atau dalam hal ini, mengurangi nilai Y dari posisi peluru.
- Dalam implementasinya, posisi peluru diubah dengan menggunakan setLocation(getX(), getY() - 5), yang menggeser peluru ke atas sejauh 5 piksel setiap kali metode act() dipanggil.

Metode disappear():

- Metode ini bertanggung jawab untuk menghapus peluru dari dunia permainan jika telah melewati batas tertentu.
- Jika posisi Y dari peluru sama dengan 0 (mungkin merupakan batas atas layar), maka objek Fire akan dihapus dari dunia permainan menggunakan getWorld().removeObject(this).

```
public class GameOver extends Actor{
    public void act()
    {
    }
}
```

Pendefinisian Kelas:

- Kelas ini memiliki struktur kosong di dalam metode act(). Karena tidak ada implementasi di dalam metode act(), kelas ini tidak melakukan tindakan apa pun ketika pemanggilan act() terjadi.

Kelas Turunan dari Actor:

- Dengan menjadi turunan dari kelas Actor, kelas GameOver memiliki kemampuan untuk ditampilkan di dunia permainan dalam lingkungan Greenfoot.
- Kelas Actor dalam Greenfoot biasanya digunakan untuk objek-objek yang dapat berinteraksi dan ditampilkan di dalam dunia permainan.

```

import greenfoot.*;
import java.awt.*; //import color dari java
import greenfoot.Color; //import color ke greenfoot

public class Health extends Actor{
    int darah = 20;
    int darahbarLebar = 80;
    int darahbarTinggi = 10;
    int pixelsPerDarahPoin = (int) darahbarLebar/darah;

    public Health(){
        blood();
    }

    public void act()
    {
        blood();
    }

    public void blood(){
        setImage(new GreenfootImage(darahbarLebar + 2, darahbarTinggi + 2));
        getImage().setColor(Color.BLACK);
        getImage().drawRect(0, 0, darahbarLebar + 1, darahbarTinggi + 1);
        getImage().setColor(Color.RED);
        getImage().fillRect(1, 1, darah*pixelsPerDarahPoin, darahbarLebar);
    }

    public void loseBlood(){
        darah--;
        kalah();
        blood();
    }

    public void act()
    {
        blood();
    }

    public void blood(){
        setImage(new GreenfootImage(darahbarLebar + 2, darahbarTinggi + 2));
        getImage().setColor(Color.BLACK);
        getImage().drawRect(0, 0, darahbarLebar + 1, darahbarTinggi + 1);
        getImage().setColor(Color.RED);
        getImage().fillRect(1, 1, darah*pixelsPerDarahPoin, darahbarLebar);
    }

    public void loseBlood(){
        darah--;
        kalah();
    }

    public void kalah(){
        if(darah == 0){
            getWorld().addObject(new GameOver(), 300, 300);
            Greenfoot.playSound("gameover1.mp3");
            Greenfoot.stop();
        }
    }
}

```

Variabel Kelas:

- darah: Variabel integer yang menunjukkan jumlah kesehatan atau darah pemain.
- darahbarLebar dan darahbarTinggi: Variabel integer yang menunjukkan lebar dan tinggi dari batang darah yang akan ditampilkan.
- pixelsPerDarahPoin: Variabel yang menghitung jumlah piksel yang mewakili setiap poin darah pada batang darah.

Konstruktor Health():

- Konstruktor ini dipanggil saat objek Health dibuat.
- Konstruktor ini memanggil metode blood() untuk menginisialisasi tampilan batang darah.

Metode act():

- Metode act() dipanggil secara berkala oleh lingkungan permainan.
- Di dalam kode ini, metode act() memanggil kembali metode blood() untuk memperbarui tampilan batang darah.

Metode blood():

- Metode ini mengatur tampilan dari batang darah.
- Pertama-tama, sebuah GreenfootImage baru dengan ukuran sesuai dengan lebar dan tinggi batang darah dibuat.
- Warna hitam digunakan untuk menggambar kontur atau tepi batang darah.
- Warna merah digunakan untuk mengisi bagian batang darah sesuai dengan jumlah darah yang tersisa.

Metode loseBlood():

- Metode ini dipanggil ketika pemain kehilangan satu poin darah.
- Jumlah darah dikurangi satu dan kemudian dipanggil metode kalah().

Metode kalah():

- Metode ini memeriksa apakah darah telah habis (0).
- Jika darah mencapai 0, maka permainan dihentikan (Greenfoot.stop()) dan objek GameOver ditambahkan ke dunia permainan pada koordinat (300, 300).
- Selain itu, suara game over diputar menggunakan Greenfoot.playSound().

```
import greenfoot.*;
import java.awt.*; //import color dari java
import greenfoot.Color; //import color ke greenfoot

public class Point extends Actor{
    int poin = 0;
    public Point(){
        setImage(new GreenfootImage("Score : " + poin, 30, Color.RED, Color.LIGHT_GRAY));
    }

    public void act()
    {
        setImage(new GreenfootImage("Score : " + poin, 30, Color.RED, Color.LIGHT_GRAY));
        menang();
    }

    public void tambahNilai(){
        poin++;
    }

    public void menang(){
        if(poin >= 20 || poin == 21){
            getWorld().addObject(new Winner(), 300, 300);
            Greenfoot.playSound("winner.wav");
            Greenfoot.stop();
        }
    }
}
```

Variabel Kelas:

- poin: Variabel integer yang menyimpan nilai skor atau poin pemain.

Konstruktur Point():

- Konstruktur ini dipanggil saat objek Point dibuat.
- Konstruktur ini menetapkan gambaran awal untuk objek Point menggunakan GreenfootImage. Gambaran ini berisi teks "Score : " diikuti dengan nilai awal dari variabel poin (dalam hal ini, 0). Ukuran teks adalah 30, dengan warna teks merah dan latar belakang abu-abu muda.

Metode act():

- Metode act() dipanggil secara berkala oleh lingkungan permainan.
- Di dalam metode ini, gambaran untuk objek Point diperbarui dengan nilai terbaru dari variabel poin yang digabungkan dengan teks "Score : ".
- Setelah perbaruan tampilan, metode menang() dipanggil untuk memeriksa apakah pemain telah mencapai nilai tertentu untuk memenangkan permainan.

Metode tambahNilai():

- Metode ini dipanggil ketika pemain mendapatkan poin atau skor.
- Nilai dari variabel poin ditambah satu setiap kali metode ini dipanggil, menambah skor pemain.

Metode menang():

- Metode ini memeriksa apakah pemain telah mencapai syarat kemenangan tertentu dalam permainan.
- Jika nilai dari variabel poin mencapai atau melebihi 20, atau jika poin sama dengan 21, maka permainan dihentikan (Greenfoot.stop()) dan objek Winner ditambahkan ke dunia permainan pada koordinat (300, 300).
- Selain itu, suara kemenangan diputar menggunakan Greenfoot.playSound().

```

import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Winner here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Winner extends Actor
{
    /**
     * Act - do whatever the Winner wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}

```

Kelas Winner:

- Kelas ini saat ini tidak memiliki logika yang diimplementasikan di dalam metode act().
- Metode act() yang ada saat ini kosong, tidak memiliki tindakan atau perilaku apa pun yang ditetapkan di dalamnya.

Deskripsi Kelas:

- Terdapat komentar yang menunjukkan bahwa kelas ini seharusnya memiliki deskripsi (jelaskan peran atau tujuan kelas ini) di dalam komentar sebelumnya.

Tujuan Kelas:

- Kelas Winner kemungkinan diciptakan untuk menangani atau menampilkan perilaku tertentu yang terkait dengan kondisi kemenangan dalam permainan.
- Biasanya, kelas semacam ini akan digunakan untuk menampilkan objek khusus atau pesan yang menandakan kemenangan saat pemain berhasil menyelesaikan permainan.