

PEMROGRAMAN BERORIENTASI OBJEK

LAPORAN UAS PBO

GAME SPACE SHOOTER



Disusun Oleh :

Renggo Subianto (2211102441098)

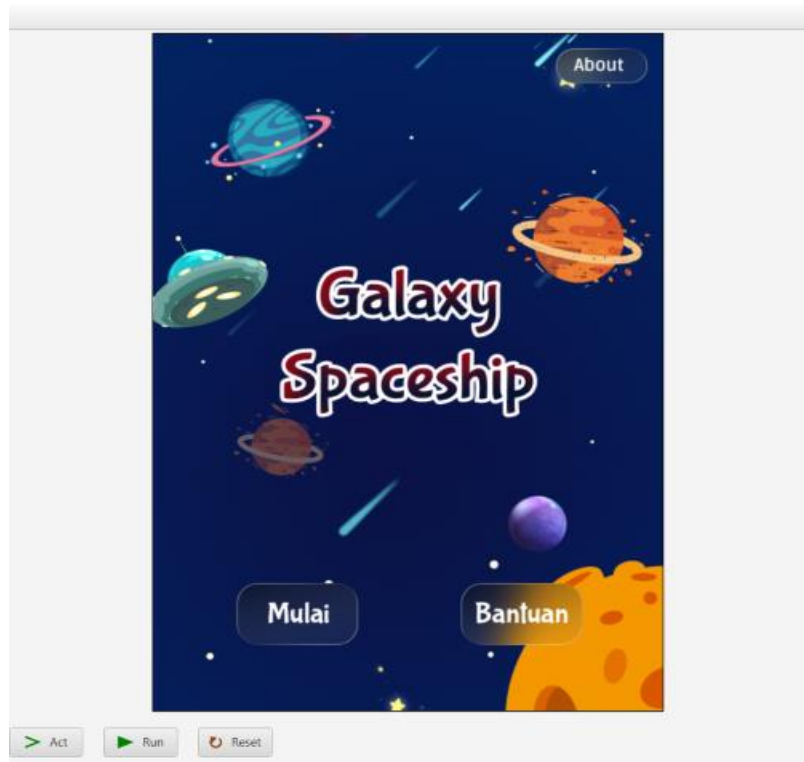
M Reza Alfiannur (221102441240)

Wahyu Saputra (2211102441108)

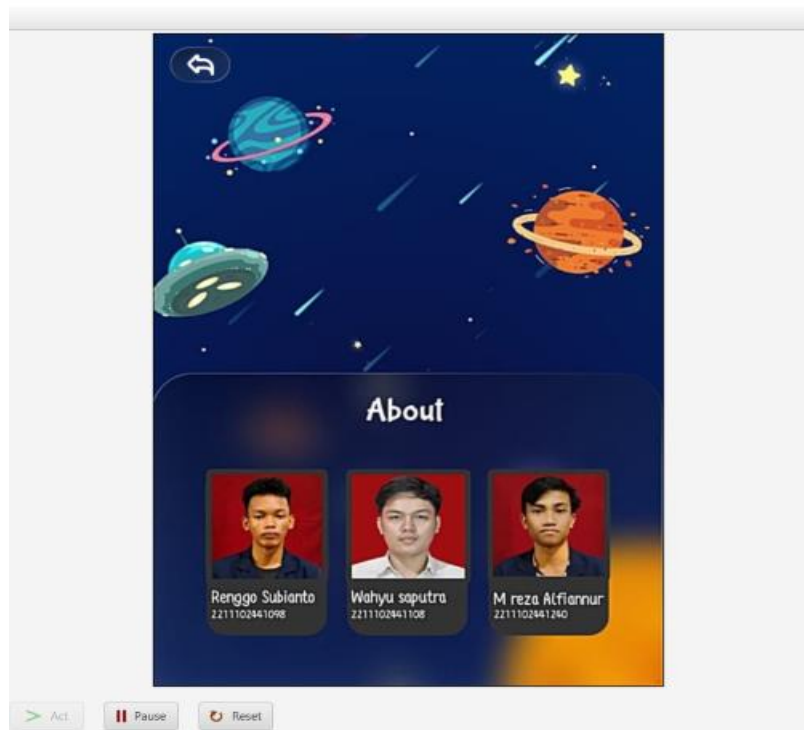
**S1 TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR
2023**

1. Tampilan Game Yang Telah Dibuat

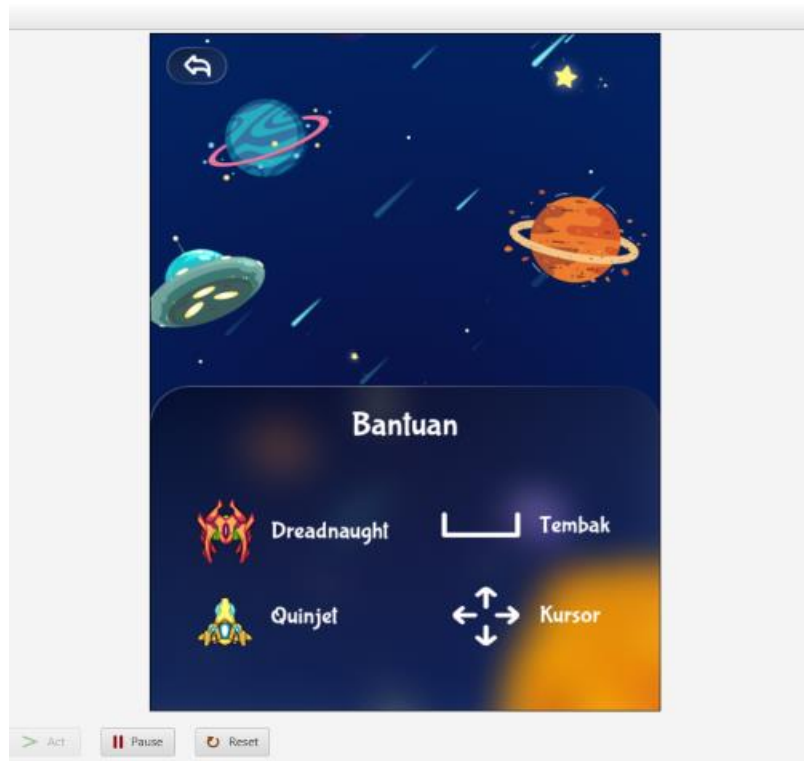
- Tampilan Beranda



- Tampilan Menu About



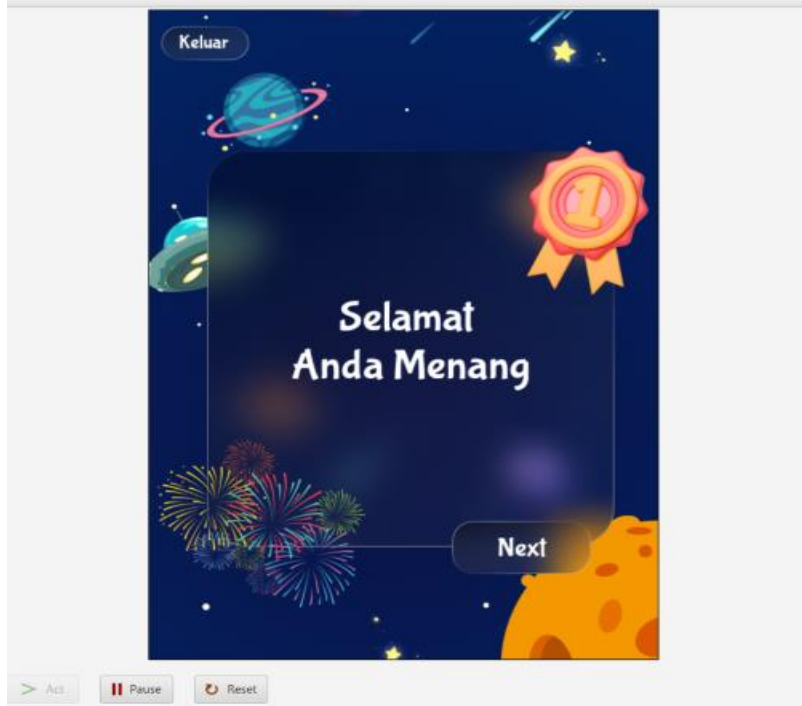
- Tampilan Menu Bantuan



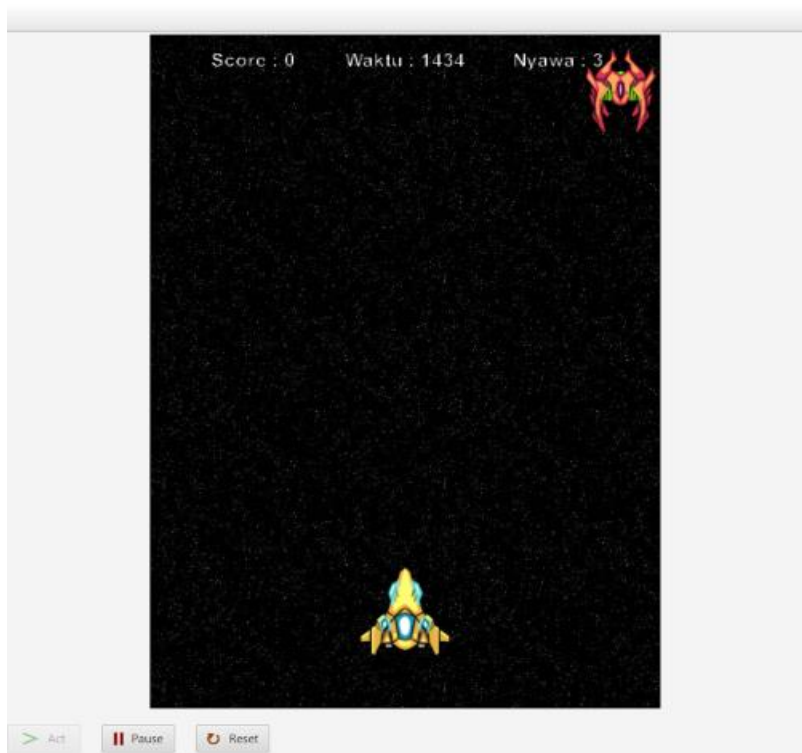
- Tampilan Menu Mulai / Map1



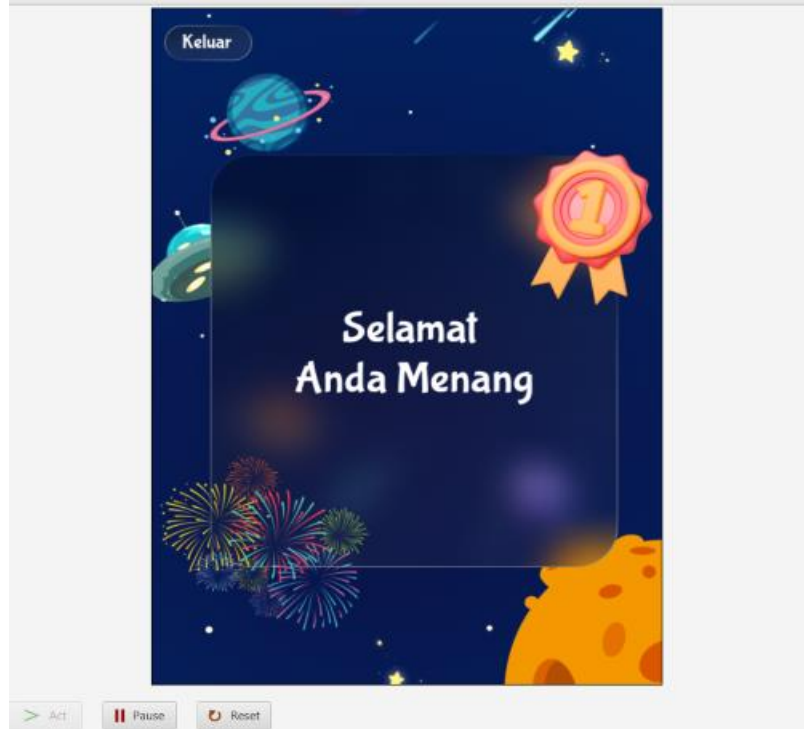
- Tampilan NextLevel



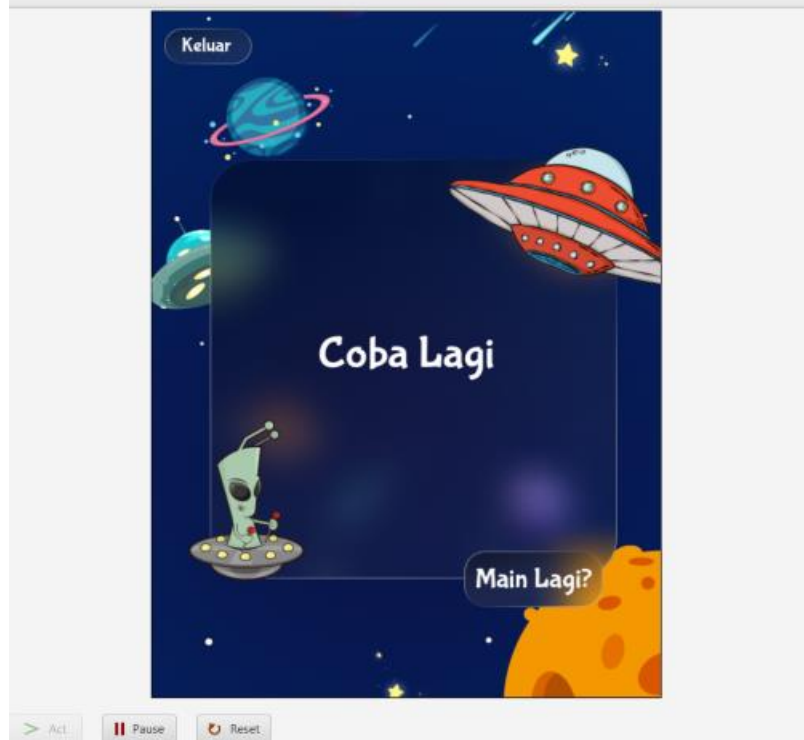
- Tampilan Map2



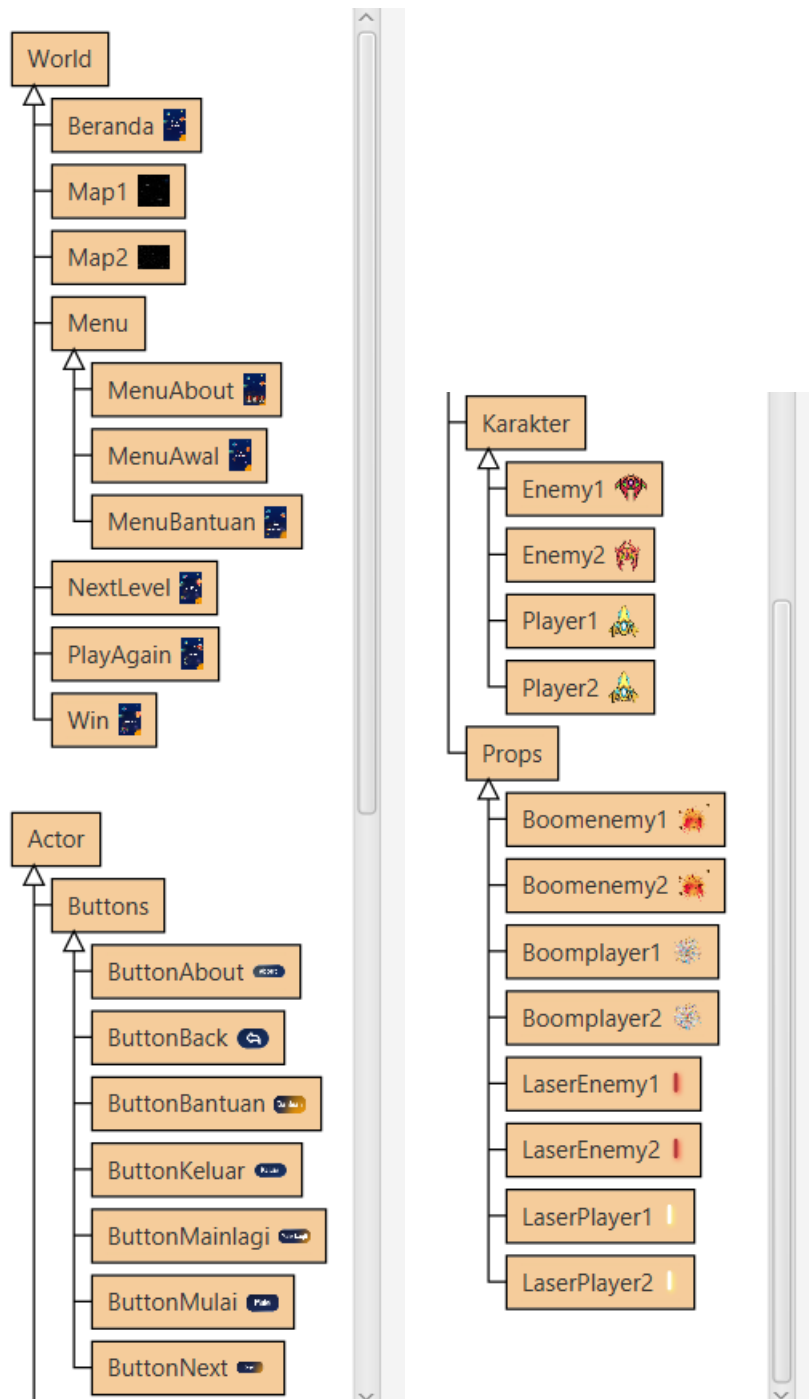
- Tampilan Menang



- Tampilan Kalah

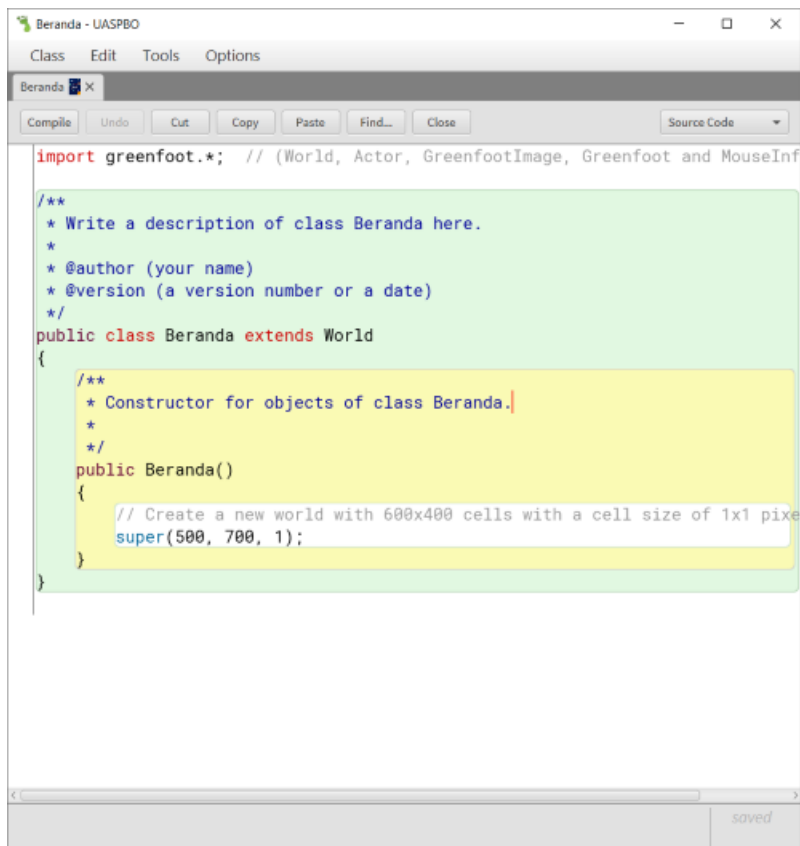


2. Class-Class yang terdapat pada Game yang telah dibuat pada Greenfoot



3. Penjelasan Tentang Class-Class Game yang terdapat pada Greenfoot

- Class Beranda



Kelas Beranda ini adalah kelas dalam framework Greenfoot yang mendefinisikan dunia awal atau layar beranda dalam permainan atau simulasi.

Yang memiliki Konstruktork untuk membuat dunia baru dengan ukuran 500x700 sel dan ukuran sel 1x1 piksel, digunakan sebagai dunia awal atau layar beranda dalam permainan Greenfoot.

Tujuannya adalah sebagai kelas utama yang menjadi titik awal permainan dan membentuk dasar lingkungan grafis di mana elemen-elemen permainan atau simulasi dapat ditambahkan.

Kelas ini digunakan untuk menginisialisasi dunia permainan dan mungkin menyertakan elemen-elemen seperti pemain awal, objek-objek lingkungan, atau instruksi awal dalam permainan.

- Class Map1

```
Map1 - UAS PBO
Class Edit Tools Options
Map1 x
Compile Undo Cut Copy Paste Find... Close Source Code
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class MyWorld here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Map1 extends World
{
    private int score;
    private int nyawa = 3;
    private int time;
    GreenfootSound backgroundMusic = new GreenfootSound("sound.wav");

    /**
     * Constructor for objects of class MyWorld.
     *
     */
    public Map1()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(500, 700, 1);
        tampilnyawa();
        tampilscore();
        showTime();
        time = 1000;
        addObject(new Player1(), 250, 600);
        backgroundMusic.play();
    }

    //menampilkan terjadinya update nyawa ketika tertembak
    public void updatenyawa(int point)
    {
        nyawa = nyawa + point;
        tampilnyawa();

        if(nyawa == 0)
        {
            Greenfoot.setWorld(new PlayAgain());
        }
    }

    //menampilkan nyawa
    public void tampilnyawa()
    {
        showText("Nyawa : " + nyawa, 400, 25);
    }

    //menampilkan terjadinya perubahan score ketika berhasil menembak musuh
    public void addscore(int points)
    {
        score = score + points;
        tampilscore();
    }
}
```



```
//menampilkan score
private void tampilScore()
{
    showText("Score : " + score, 100, 25);
}

//menampilkan waktu yang terus berkurang
private void countTime()
{
    time --;
    showTime();
    if (time == 0)
    {
        Greenfoot.setWorld(new NextLevel());
    }
}

//menampilkan waktu
private void showTime()
{
    showText("Waktu : " + time, 250, 25);
}

//menampilkan musuh yang keluar
public void act()
{
    if (Greenfoot.getRandomNumber (100) < 2)
    {
        addObject(new Enemy1(), Greenfoot.getRandomNumber(599), 500);
    }
    countTime();
}
}
```

Class compiled - no syntax errors saved

Kelas Map1 merupakan bagian dari permainan Greenfoot. Berikut adalah penjelasan fungsionalitas dan komponen-komponennya.

Variabel Kelas: Score untuk menyimpan nilai skor permainan, Nyawa untuk menyimpan jumlah nyawa pemain, Time untuk menyimpan waktu tersisa dalam permainan dan BackgroundMusic untuk objek suara untuk musik latar belakang.

Konstruktor:

- Membuat dunia baru dengan ukuran dan karakteristik tertentu.
- Menampilkan informasi nyawa, skor, dan waktu.
- Menambahkan pemain ke dunia.
- Memainkan musik latar belakang.

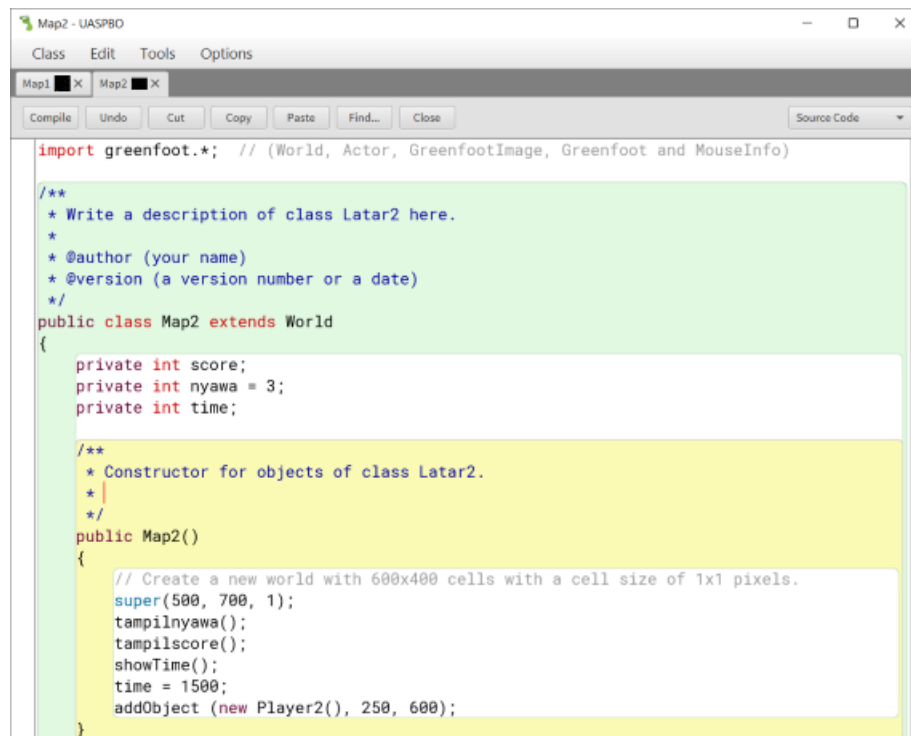
Metode Tambahan:

- updatenyawa(int point): Menambah atau mengurangi nyawa dan memperbarui tampilan. Pindah ke dunia baru jika nyawa habis.
- tampilnyawa(): Menampilkan jumlah nyawa.
- addscore(int points): Menambah nilai skor dan memperbarui tampilan.

-tampilScore(): Menampilkan nilai skor.
-countTime(): Mengurangi waktu dan memperbarui tampilan. Pindah ke dunia baru jika waktu habis.
-showTime(): Menampilkan sisa waktu.
-act(): Dipanggil setiap iterasi dunia. Menambahkan musuh ke dunia secara acak dan memproses waktu.

Kelas ini menyusun elemen-elemen dasar permainan, termasuk pemain, musuh, skor, waktu, dan musik latar belakang. Mekanisme untuk mengelola nyawa, skor, dan waktu, serta perpindahan ke level berikutnya telah diimplementasikan.

- Class Map2



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

/**
 * Write a description of class Latar2 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Map2 extends World
{
    private int score;
    private int nyawa = 3;
    private int time;

    /**
     * Constructor for objects of class Latar2.
     *
     */
    public Map2()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(500, 700, 1);
        tampilnyawa();
        tampilScore();
        showTime();
        time = 1500;
        addObject (new Player2(), 250, 600);
    }
}
```

```
//menampilkan terjadinya update nyawa ketika tertembak
public void updatenyawa(int point)
{
    nyawa = nyawa + point;
    tampilnyawa();

    if(nyawa == 0)
    {
        Greenfoot.setWorld(new PlayAgain());
    }
}

//menampilkan nyawa
public void tampilnyawa()
{
    showText("Nyawa : " + nyawa, 400, 25);
}

//menampilkan terjadinya perubahan score ketika berhasil menembak musuh
public void addscore(int points)
{
    score = score + points;
    tampilscore();
}

//menampilkan score
private void tampilscore()
{
    showText("Score : " + score, 100, 25);
}

}

//menampilkan waktu yang terus berkurang
private void countTime()
{
    time --;
    showTime();
    if (time == 0)
    {
        Greenfoot.setWorld(new Win());
    }
}

//menampilkan waktu
private void showTime()
{
    showText("Waktu : " + time, 250, 25);
}

//menampilkan musuh yang keluar
public void act()
{
    if (Greenfoot.getRandomNumber (100) < 5)
    {
        addObject(new Enemy2(), Greenfoot.getRandomNumber(599), 500);
    }
    countTime();
}
}
```

Class compiled - no syntax errors

saved

Kelas Map2 memiliki fungsionalitas yang mirip dengan Map1 namun dengan beberapa perbedaan. Berikut adalah penjelasannya.

Variabel Kelas:

-score: Menyimpan nilai skor permainan.

- nyawa: Menyimpan jumlah nyawa pemain.
- time: Menyimpan waktu tersisa dalam permainan.

Konstruktor:

- Membuat dunia baru dengan ukuran dan karakteristik tertentu.
- Menampilkan informasi nyawa, skor, dan waktu.
- Menambahkan pemain (Player2) ke dunia pada posisi tertentu.

Metode Tambahan:

- updatenyawa(int point): Menambah atau mengurangi nyawa dan memperbarui tampilan. Pindah ke dunia baru jika nyawa habis.
- tampilnyawa(): Menampilkan jumlah nyawa.
- addscore(int points): Menambah nilai skor dan memperbarui tampilan.
- tampilscore(): Menampilkan nilai skor.
- countTime(): Mengurangi waktu dan memperbarui tampilan. Pindah ke dunia baru jika waktu habis.
- showTime(): Menampilkan sisa waktu.
- act(): Dipanggil setiap iterasi dunia. Menambahkan musuh (Enemy2) ke dunia secara acak dan memproses waktu.

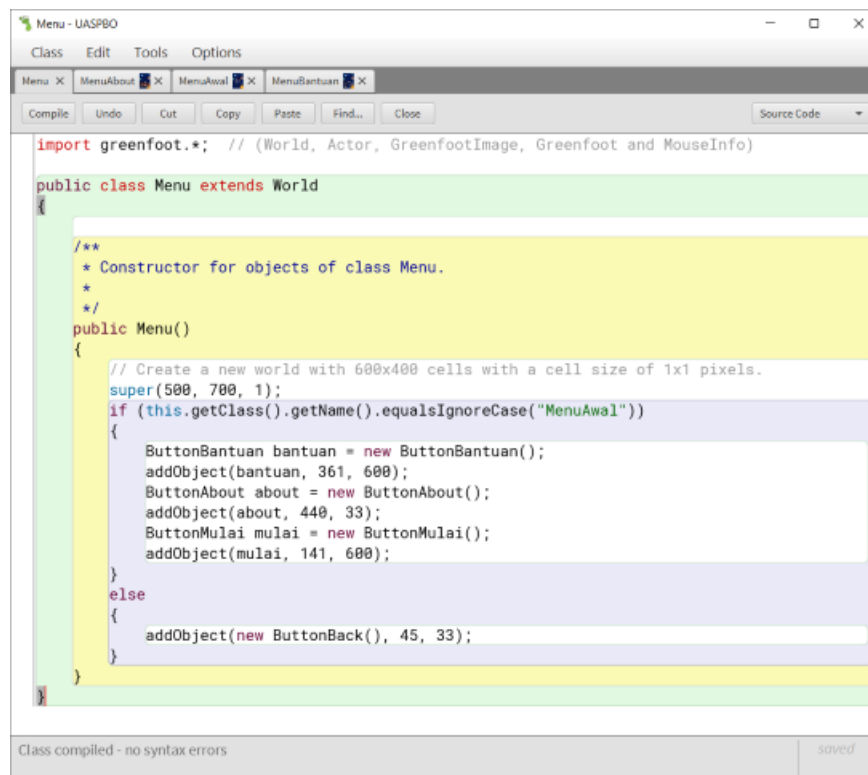
Perbedaan dengan Map1:

Pada konstruktor, objek pemain yang ditambahkan adalah Player2.
Waktu awal permainan (time) diatur menjadi 1500.
Pada metode act(), kemungkinan musuh muncul dengan nilai kurang sering ($\text{Greenfoot.getRandomNumber}(100) < 5$).

Kelas Map2 merupakan level berikutnya dalam permainan, dengan karakteristik dan kesulitan yang berbeda dari level sebelumnya (Map1).

- Class Menu dan Turunannya (MenuAbout, MenuAwal dan MenuBantuan)

-Class Menu



```

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

public class Menu extends World
{
    /**
     * Constructor for objects of class Menu.
     */
    public Menu()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(500, 700, 1);
        if (this.getClass().getName().equalsIgnoreCase("MenuAwal"))
        {
            ButtonBantuan bantuan = new ButtonBantuan();
            addObject(bantuan, 361, 600);
            ButtonAbout about = new ButtonAbout();
            addObject(about, 440, 33);
            ButtonMulai mulai = new ButtonMulai();
            addObject(mulai, 141, 600);
        }
        else
        {
            addObject(new ButtonBack(), 45, 33);
        }
    }
}

```

Class compiled - no syntax errors

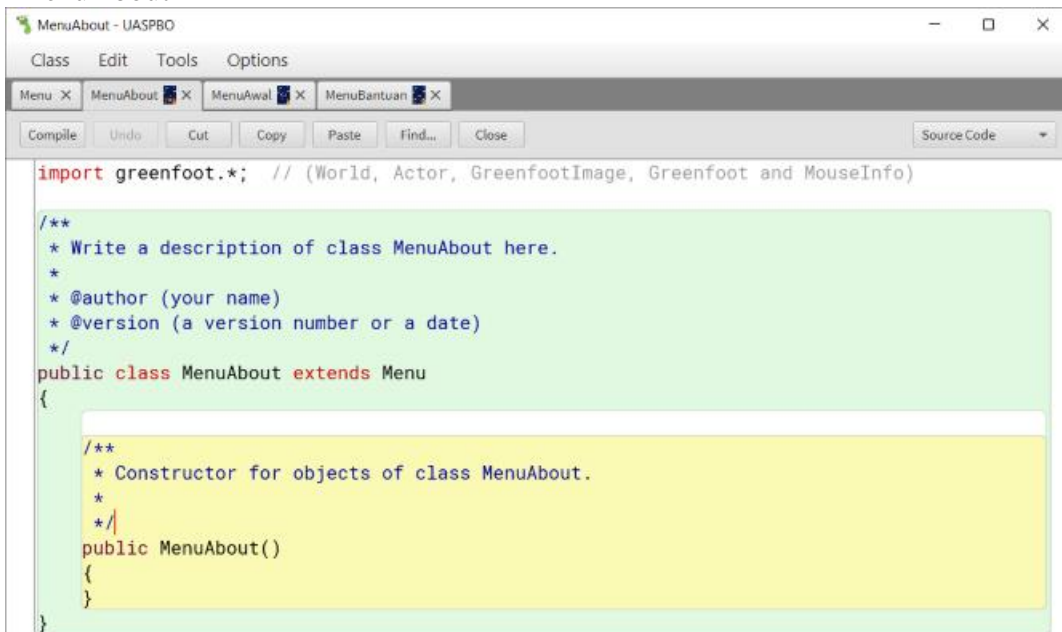
Kelas Menu adalah kelas yang mendefinisikan layar menu dalam permainan. Menyediakan tampilan menu di awal permainan atau menu tambahan, tergantung pada kelas yang memanggilnya. Memfasilitasi navigasi atau pilihan bagi pengguna untuk memulai permainan, melihat bantuan, melihat informasi anggota kelompok (About), atau kembali ke layar sebelumnya.

Konstruktor:

- Membuat dunia baru dengan ukuran dan karakteristik tertentu.
- Mengecek nama kelas saat ini dengan menggunakan `this.getClass().getName()`.
- Jika kelas ini adalah MenuAwal (dengan membandingkan secara tidak peka terhadap - huruf besar/kecil), maka menambahkan tiga tombol ke dunia: ButtonBantuan, ButtonAbout, dan ButtonMulai.
- Jika kelas ini bukan MenuAwal, maka menambahkan satu tombol ButtonBack ke dunia.

Kelas ini memanfaatkan perbandingan nama kelas untuk menentukan jenis tampilan menu yang diperlukan. Jika kelas yang memanggil adalah MenuAwal, maka tiga tombol tambahan akan ditambahkan ke dunia. Jika bukan, hanya satu tombol kembali yang ditambahkan.

-MenuAbout

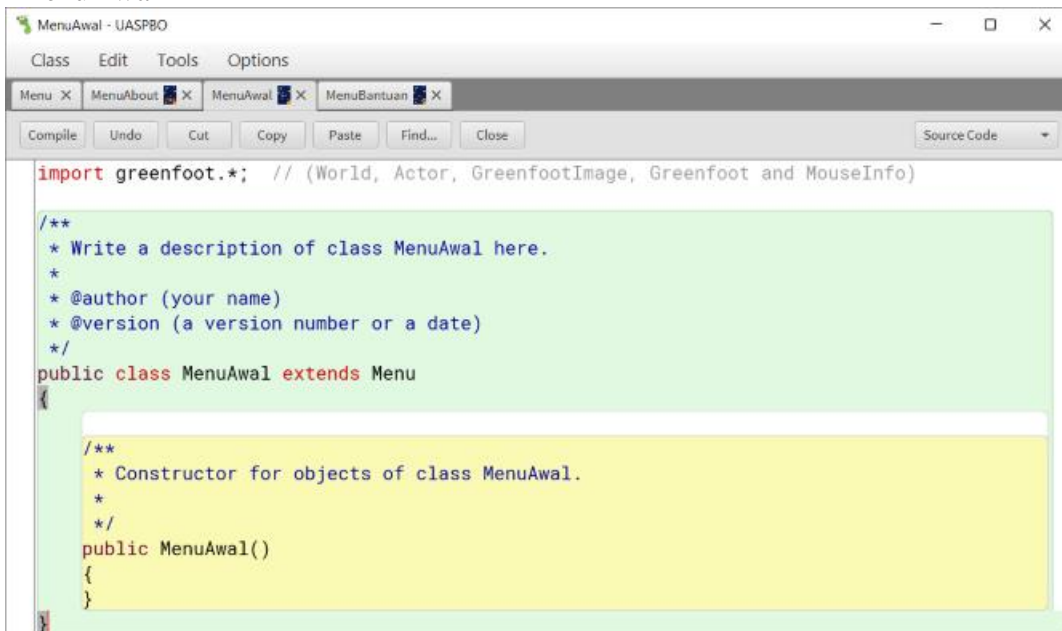


The screenshot shows the MenuAbout - UASPB0 IDE window. The menu bar includes Class, Edit, Tools, and Options. The tab bar shows Menu, MenuAbout, MenuAwal, and MenuBantuan. The toolbar includes Compile, Undo, Cut, Copy, Paste, Find..., and Close. The Source Code dropdown is visible. The code editor displays the following Java code:

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

/**
 * Write a description of class MenuAbout here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class MenuAbout extends Menu
{
    /**
     * Constructor for objects of class MenuAbout.
     *
     */
    public MenuAbout()
    {
    }
}
```

-MenuAwal

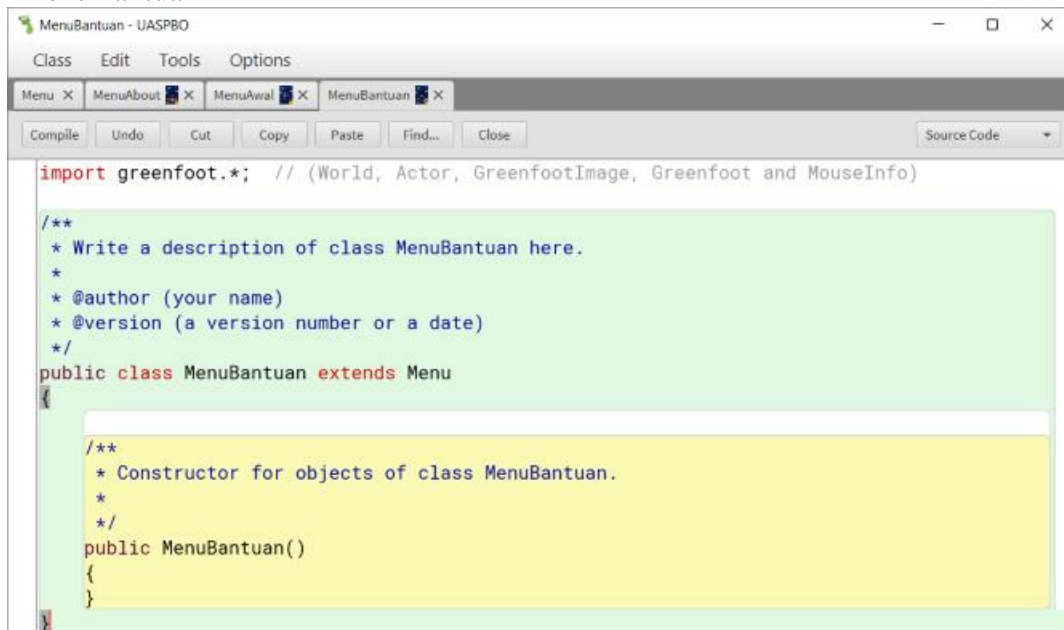


The screenshot shows the MenuAwal - UASPB0 IDE window. The menu bar includes Class, Edit, Tools, and Options. The tab bar shows Menu, MenuAbout, MenuAwal, and MenuBantuan. The toolbar includes Compile, Undo, Cut, Copy, Paste, Find..., and Close. The Source Code dropdown is visible. The code editor displays the following Java code:

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

/**
 * Write a description of class MenuAwal here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class MenuAwal extends Menu
{
    /**
     * Constructor for objects of class MenuAwal.
     *
     */
    public MenuAwal()
    {
    }
}
```

-MenuBantuan



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

/**
 * Write a description of class MenuBantuan here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class MenuBantuan extends Menu
{
    /**
     * Constructor for objects of class MenuBantuan.
     *
     */
    public MenuBantuan()
    {
    }
}
```

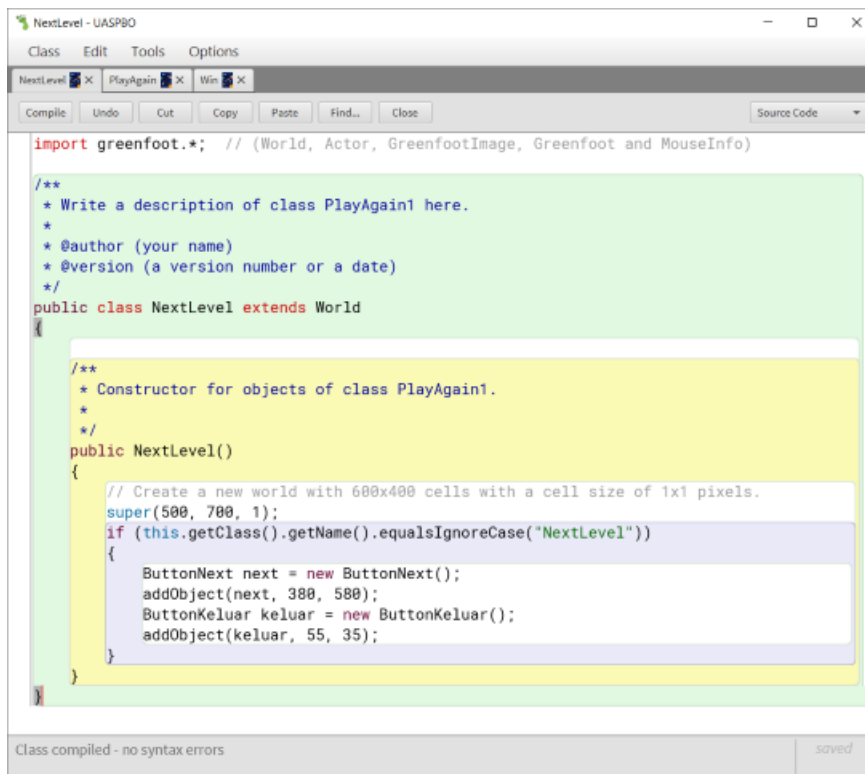
Subkelas MenuAbout, MenuAwal, dan MenuBantuan memiliki penjelasan yang serupa karena semuanya merupakan subkelas dari kelas Menu. Yang menunjukkan bahwa ketiganya mewarisi fungsionalitas dasar dari kelas Menu.

Subclass MenuAbout, MenuAwal, Menu Bantuan:

- Ketiga kelas ini mewarisi fungsionalitas dasar dari kelas Menu.
- Dirancang untuk menetapkan tampilan spesifik terkait masing-masing bagian menu dalam permainan atau simulasi.
- Menambahkan perilaku atau elemen tambahan yang sesuai dengan konteksnya (MenuAbout untuk informasi tentang permainan, MenuAwal untuk layar awal permainan, dan MenuBantuan untuk menu bantuan).

Dengan menggunakan pendekatan ini, kita dapat melihat bahwa struktur dasar dan tujuan fungsi dari setiap kelas sub adalah serupa, dengan setiap kelas sub dapat memodifikasi atau menambahkan perilaku sesuai dengan kebutuhan spesifiknya. Pendekatan ini mendemonstrasikan penggunaan pewarisan untuk mengorganisir dan membangun hierarki kelas dalam proyek Greenfoot.

- Class NextLevel



```

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

/**
 * Write a description of class PlayAgain1 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class NextLevel extends World
{
    /**
     * Constructor for objects of class PlayAgain1.
     *
     */
    public NextLevel()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(500, 700, 1);
        if (this.getClass().getName().equalsIgnoreCase("NextLevel"))
        {
            ButtonNext next = new ButtonNext();
            addObject(next, 380, 580);
            ButtonKeluar keluar = new ButtonKeluar();
            addObject(keluar, 55, 35);
        }
    }
}

```

Class compiled - no syntax errors

KelasNextLevel mendefinisikan dunia atau layar untuk menampilkan opsi setelah menyelesaikan level tertentu, dalam hal ini, memproyeksikan layar untuk melanjutkan ke level berikutnya.

Menambahkan tombol "Next" (ButtonNext) untuk memungkinkan pemain melanjutkan ke level selanjutnya.

Menambahkan tombol "Keluar" (ButtonKeluar) untuk memberikan opsi untuk keluar dari permainan atau kembali ke menu awal.

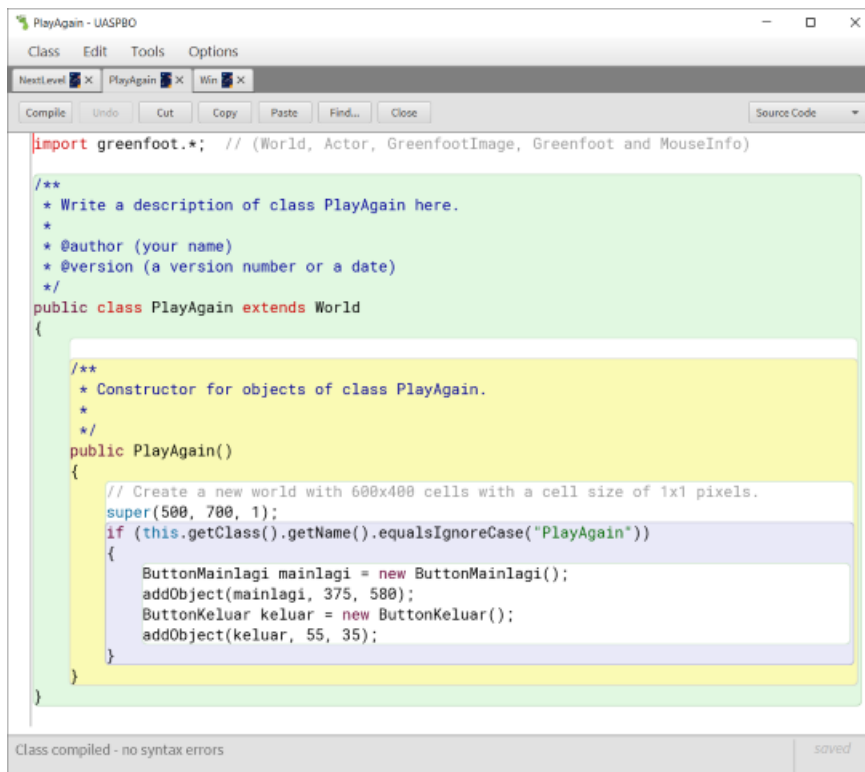
Konstruktor:

- Membuat dunia baru dengan ukuran dan karakteristik tertentu.

- Jika kelas yang memanggil adalah NextLevel, maka menambahkan dua tombol ke dunia: ButtonNext dan ButtonKeluar.

Dengan menggunakan pendekatan ini, kelas NextLevel memberikan antarmuka yang memungkinkan pemain untuk memilih untuk melanjutkan ke level berikutnya atau keluar dari permainan. Seluruh struktur kelas ini berfungsi sebagai bagian dari logika alur permainan dan memberikan interaksi antara pemain dan permainan dalam konteks Greenfoot.

- Class PlayAgain



```

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

/**
 * Write a description of class PlayAgain here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class PlayAgain extends World
{
    /**
     * Constructor for objects of class PlayAgain.
     *
     */
    public PlayAgain()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(500, 700, 1);
        if (this.getClass().getName().equalsIgnoreCase("PlayAgain"))
        {
            ButtonMainlagi mainlagi = new ButtonMainlagi();
            addObject(mainlagi, 375, 580);
            ButtonKeluar keluar = new ButtonKeluar();
            addObject(keluar, 55, 35);
        }
    }
}

```

Class compiled - no syntax errors

Kelas ini mendefinisikan dunia atau layar untuk menampilkan opsi setelah pemain menyelesaikan level atau tugas tertentu. Dalam hal ini, menampilkan layar untuk memulai permainan lagi setelah pemain kalah atau menyelesaikan suatu tugas. Menambahkan tombol "Main Lagi" (ButtonMainlagi) untuk memungkinkan pemain memulai permainan dari awal. Menambahkan tombol "Keluar" (ButtonKeluar) untuk memberikan opsi untuk keluar dari permainan atau kembali ke menu awal.

Konstruktor:

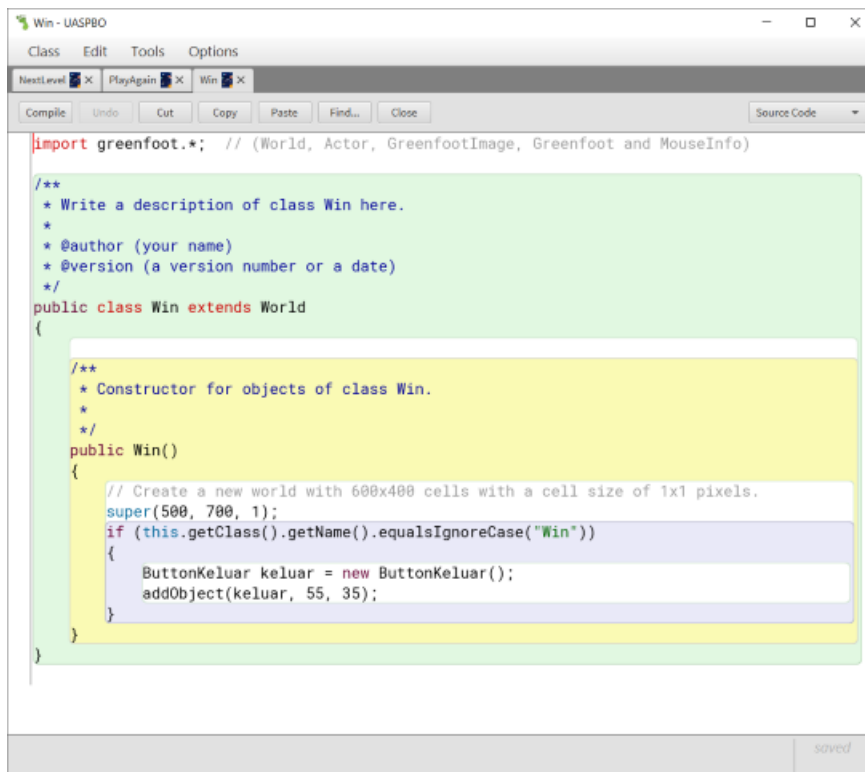
- Membuat dunia baru dengan ukuran dan karakteristik tertentu.

Jika kelas yang memanggil adalah PlayAgain, maka menambahkan dua tombol ke dunia:

- ButtonMainlagi dan ButtonKeluar.

Dengan menggunakan pendekatan ini, kelas PlayAgain memberikan antarmuka yang memungkinkan pemain untuk memilih untuk memulai permainan lagi atau keluar dari permainan setelah mencapai kondisi tertentu. Seluruh struktur kelas ini berfungsi sebagai bagian dari logika alur permainan dan memberikan interaksi antara pemain dan permainan.

- Class Win



```

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

/**
 * Write a description of class Win here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Win extends World
{
    /**
     * Constructor for objects of class Win.
     *
     */
    public Win()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(500, 700, 1);
        if (this.getClass().getName().equalsIgnoreCase("Win"))
        {
            ButtonKeluar keluar = new ButtonKeluar();
            addObject(keluar, 55, 35);
        }
    }
}

```

Kelas Win mewakili dunia atau layar yang muncul ketika pemain berhasil menyelesaikan permainan, mencapai tujuan tertentu, atau meraih kemenangan.

Kelas ini mendefinisikan dunia atau layar untuk menampilkan informasi atau opsi setelah pemain berhasil menyelesaikan permainan atau mencapai tujuan tertentu.

Dalam hal ini, menambahkan tombol "Keluar" (ButtonKeluar) untuk memberikan opsi kepada pemain untuk keluar dari permainan atau kembali ke menu awal.

Konstruktor:

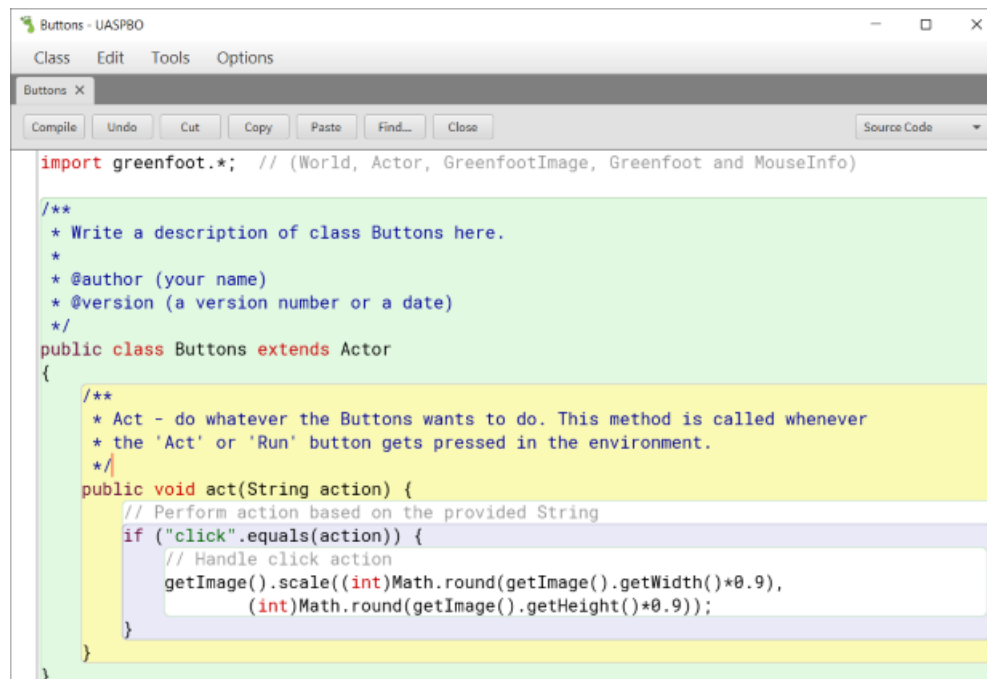
- Membuat dunia baru dengan ukuran dan karakteristik tertentu.

- Jika kelas yang memanggil adalah Win, maka menambahkan tombol "Keluar" ke dunia.

Kelas Win memberikan antarmuka yang memungkinkan pemain untuk memilih keluar dari permainan setelah mencapai kemenangan. Keseluruhan struktur kelas ini berfungsi sebagai bagian dari logika alur permainan dan memberikan interaksi antara pemain dan permainan.

- Class Buttons dan Class-Class Turunannya

-Class Buttons

A screenshot of a code editor window titled "Buttons - UASPB0". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The code editor shows the following code:

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

/**
 * Write a description of class Buttons here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Buttons extends Actor
{
    /**
     * Act - do whatever the Buttons wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act(String action) {
        // Perform action based on the provided String
        if ("click".equals(action)) {
            // Handle click action
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
                (int)Math.round(getImage().getHeight()*0.9));
        }
    }
}
```

Kelas Buttons merupakan kelas dasar yang mewarisi dari kelas Actor dalam Greenfoot.

Kelas ini bertindak sebagai kelas dasar untuk semua tombol (buttons) dalam Greenfoot.

Merupakan subkelas dari Actor, sehingga dapat ditempatkan di dunia dan berinteraksi dengan elemen-elemen lainnya.

Deklarasi Kelas:

Kelas Buttons dideklarasikan dan mewarisi sifat dari kelas Actor. Ini berarti objek yang dibuat dari kelas Buttons dapat ditambahkan ke dalam dunia Greenfoot dan mewarisi perilaku dasar dari kelas Actor.

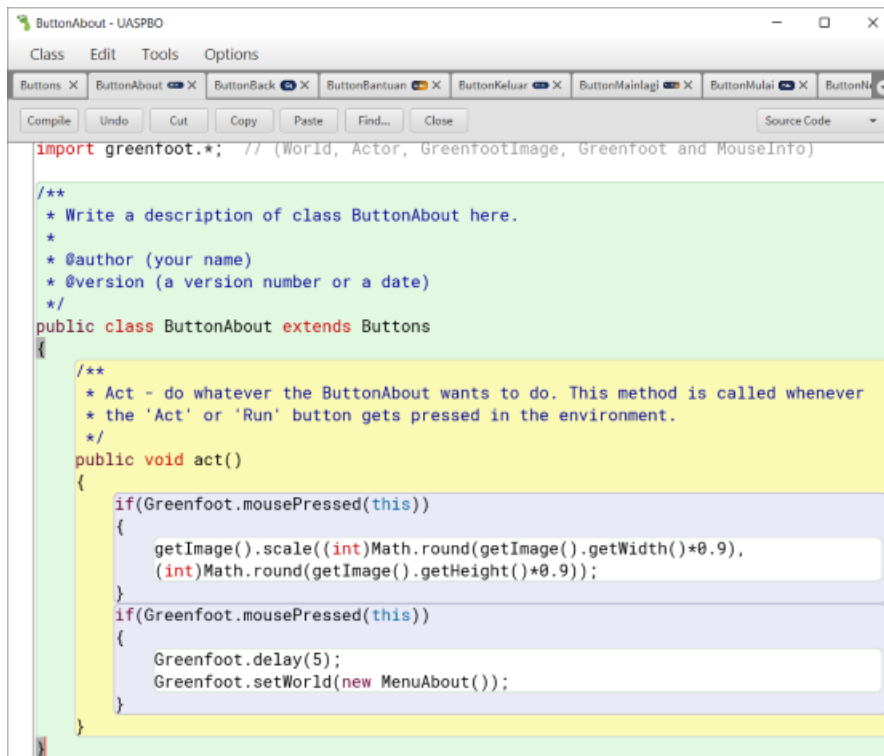
Metode Act:

-Metode act diimplementasikan untuk menanggapi suatu aksi berdasarkan nilai parameter action yang diberikan.

-Jika nilai action sama dengan string "click", maka gambar tombol akan diperkecil (diskalakan) sehingga lebar dan tingginya menjadi 90% dari nilai aslinya.

Dengan menggunakan kelas ini, Anda dapat membuat tombol-tombol dalam permainan Greenfoot dan mengatur respons terhadap tindakan tertentu, seperti mengklik tombol untuk mengubah ukuran gambar tombol.

-Class ButtonAbout

A screenshot of a Java IDE window titled "ButtonAbout - UASPBO". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a toolbar with buttons for "Buttons", "ButtonAbout", "ButtonBack", "ButtonBantuan", "ButtonKeluar", "ButtonMainlagi", "ButtonMulai", and "ButtonN". There is also a "Compile" button and a "Find..." button. The main area displays the source code for the ButtonAbout class. The code starts with an import statement for greenfoot.*. It includes a multi-line comment describing the class and its author/version. The class is defined as public class ButtonAbout extends Buttons. Inside the class, there is a comment for the act() method, followed by the method signature public void act(). The method body contains two if statements: the first checks for a mouse press and scales the image to 90% of its original size; the second checks for a mouse press and delays for 5 units of time before setting the world to a new MenuAbout instance.

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

/**
 * Write a description of class ButtonAbout here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ButtonAbout extends Buttons
{
    /**
     * Act - do whatever the ButtonAbout wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.mousePressed(this))
        {
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
                (int)Math.round(getImage().getHeight()*0.9));
        }
        if(Greenfoot.mousePressed(this))
        {
            Greenfoot.delay(5);
            Greenfoot.setWorld(new MenuAbout());
        }
    }
}
```

Kelas ButtonAbout adalah subkelas dari Buttons dalam Greenfoot dan dirancang untuk merepresentasikan tombol "About" dalam permainan.

Kelas ini bertindak sebagai representasi dari tombol "About" yang dapat diinteraksi dalam permainan.

Merupakan subkelas dari Buttons, sehingga mewarisi sifat-sifat dasar dan perilaku dari kelas dasar tersebut.

Metode Act:

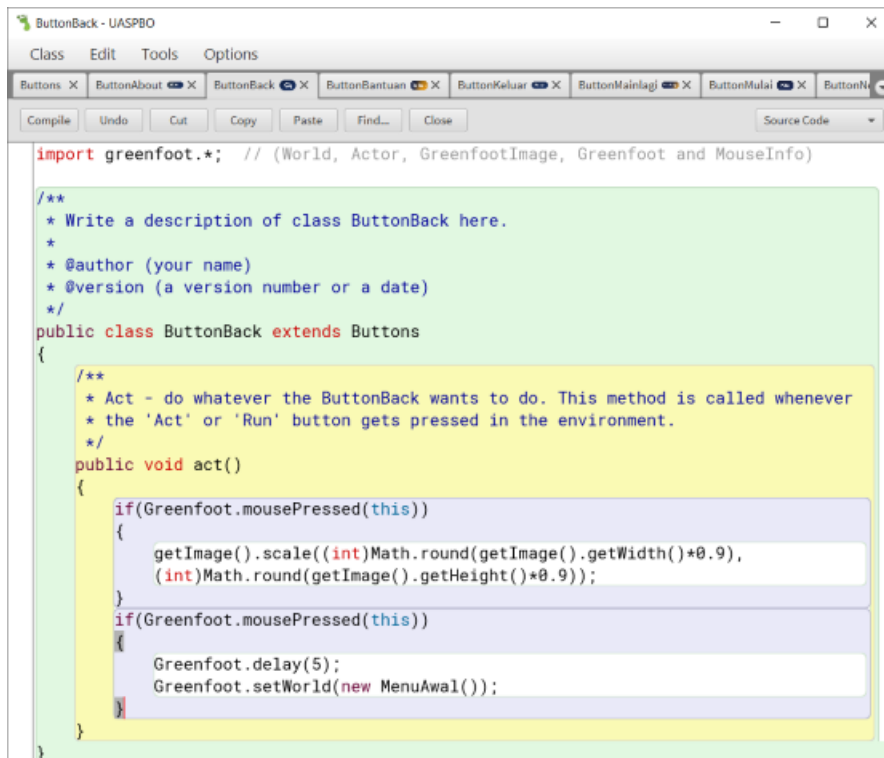
- Metode act() diimplementasikan untuk menanggapi aksi yang terjadi ketika tombol "About" ditekan.

- Jika tombol "About" ditekan, ukuran gambar tombol akan diperkecil sebesar 10% dari ukuran aslinya.

- Kemudian, setelah penundaan sebentar menggunakan Greenfoot.delay(5), dunia akan diubah menjadi MenuAbout (layar informasi tentang permainan).

Dengan menggunakan pendekatan ini, kelas ButtonAbout memberikan fungsionalitas khusus yang berkaitan dengan aksi saat tombol "About" ditekan, sambil memanfaatkan kemampuan dan sifat dasar yang diwarisi dari kelas Buttons.

-Button Back

A screenshot of a Java IDE window titled "ButtonBack - UASPBO". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a tab bar showing several open files: "Buttons", "ButtonAbout", "ButtonBack", "ButtonBantuan", "ButtonKeluar", "ButtonMainlagi", "ButtonMulai", and "ButtonN". Below the tab bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A "Source Code" dropdown menu is on the right. The main editor area displays the following Java code:

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

/**
 * Write a description of class ButtonBack here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ButtonBack extends Buttons
{
    /**
     * Act - do whatever the ButtonBack wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.mousePressed(this))
        {
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
            (int)Math.round(getImage().getHeight()*0.9));
        }
        if(Greenfoot.mousePressed(this))
        {
            Greenfoot.delay(5);
            Greenfoot.setWorld(new MenuAwal());
        }
    }
}
```

Kelas ButtonBack adalah subkelas dari Buttons dalam Greenfoot dan dirancang untuk merepresentasikan tombol "Back" (kembali) dalam permainan.

Kelas ini bertindak sebagai representasi dari tombol "Back" yang dapat diinteraksi dalam permainan.

Merupakan subkelas dari Buttons, sehingga mewarisi sifat-sifat dasar dan perilaku dari kelas dasar tersebut.

Metode Act:

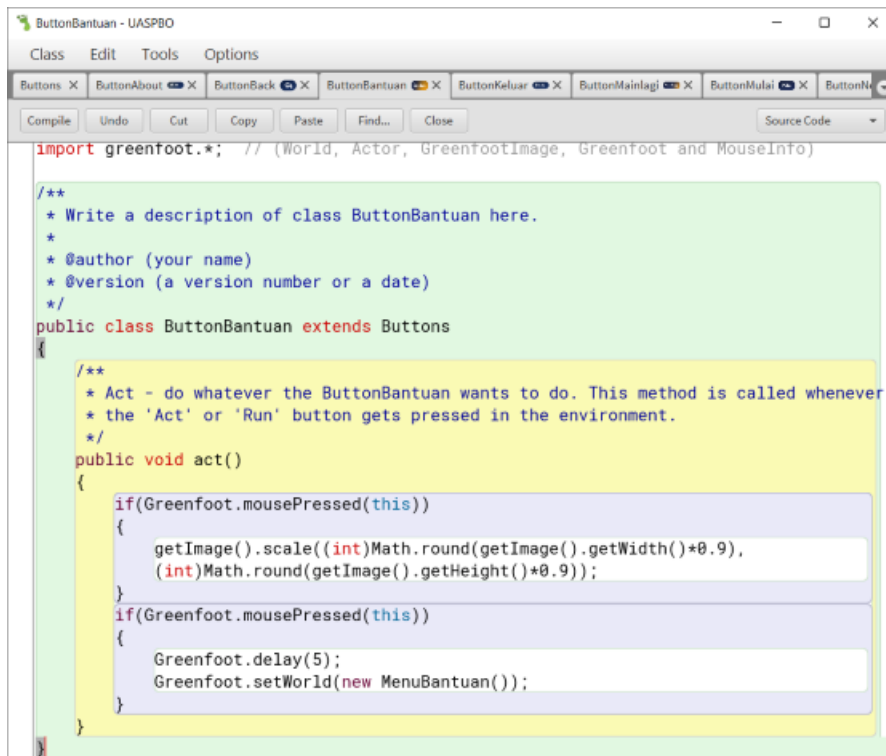
- Metode act() diimplementasikan untuk menanggapi aksi yang terjadi ketika tombol "Back" ditekan.

- Jika tombol "Back" ditekan, ukuran gambar tombol akan diperkecil sebesar 10% dari ukuran aslinya.

- Kemudian, setelah penundaan sebentar menggunakan Greenfoot.delay(5), dunia akan diubah menjadi MenuAwal (layar menu awal atau beranda).

Dengan menggunakan pendekatan ini, kelas ButtonBack memberikan fungsionalitas khusus yang berkaitan dengan aksi saat tombol "Back" ditekan, sambil memanfaatkan kemampuan dan sifat dasar yang diwarisi dari kelas Buttons.

-Button Bantuan

A screenshot of a Java IDE window titled "ButtonBantuan - UASPBO". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a tab bar showing several open files: "Buttons", "ButtonAbout", "ButtonBack", "ButtonBantuan" (the active file), "ButtonKeluar", "ButtonMainlagi", "ButtonMulai", and "ButtonN". Below the tab bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", "Close", and a "Source Code" dropdown. The main editor area displays the source code for the "ButtonBantuan" class. The code starts with an import statement for "greenfoot.*" and a comment. It then has a class comment and a Javadoc-style comment with @author and @version tags. The class is defined as "public class ButtonBantuan extends Buttons". Inside the class, there is a comment for the "act" method, followed by the method signature "public void act()". The method body contains two if-statements, both checking "Greenfoot.mousePressed(this)". The first if-statement calls "getImage().scale()" with rounded dimensions of 90% of the original. The second if-statement calls "Greenfoot.delay(5)" and "Greenfoot.setWorld(new MenuBantuan())".

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

/**
 * Write a description of class ButtonBantuan here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ButtonBantuan extends Buttons
{
    /**
     * Act - do whatever the ButtonBantuan wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.mousePressed(this))
        {
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
                (int)Math.round(getImage().getHeight()*0.9));
        }
        if(Greenfoot.mousePressed(this))
        {
            Greenfoot.delay(5);
            Greenfoot.setWorld(new MenuBantuan());
        }
    }
}
```

Kelas ButtonBantuan adalah subkelas dari Buttons dalam Greenfoot dan dirancang untuk merepresentasikan tombol "Bantuan" dalam permainan.

Kelas ini bertindak sebagai representasi dari tombol "Bantuan" yang dapat diinteraksi dalam permainan.

Merupakan subkelas dari Buttons, sehingga mewarisi sifat-sifat dasar dan perilaku dari kelas dasar tersebut.

Metode Act:

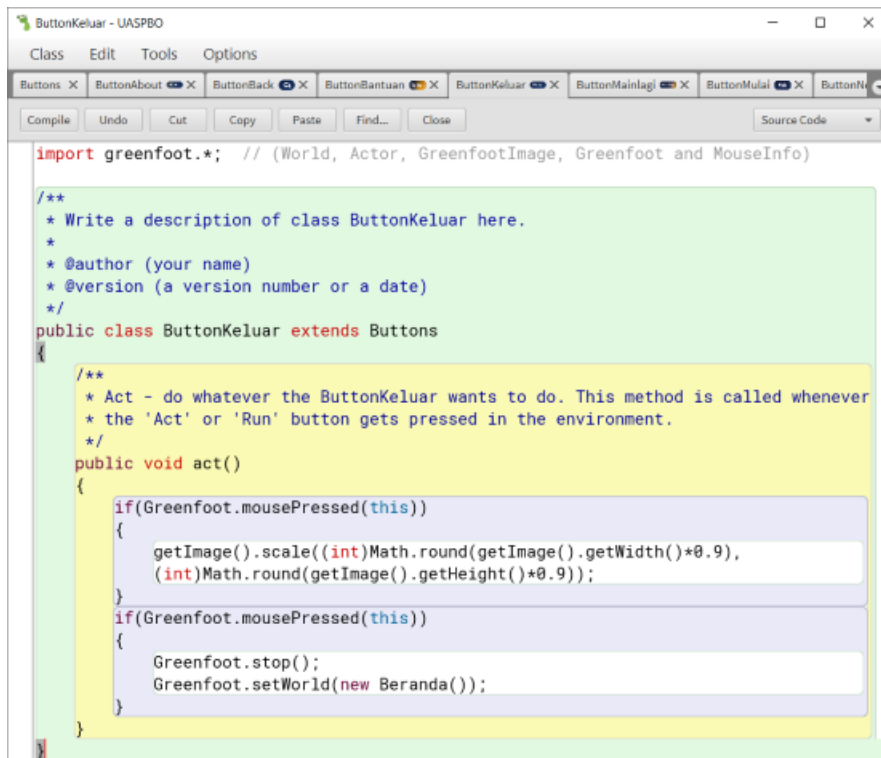
- Metode act() diimplementasikan untuk menanggapi aksi yang terjadi ketika tombol "Bantuan" ditekan.

- Jika tombol "Bantuan" ditekan, ukuran gambar tombol akan diperkecil sebesar 10% dari ukuran aslinya.

- Kemudian, setelah penundaan sebentar menggunakan Greenfoot.delay(5), dunia akan diubah menjadi MenuBantuan (layar bantuan).

Dengan menggunakan pendekatan ini, kelas ButtonBantuan memberikan fungsionalitas khusus yang berkaitan dengan aksi saat tombol "Bantuan" ditekan, sambil memanfaatkan kemampuan dan sifat dasar yang diwarisi dari kelas Buttons.

-Button Keluar

The image shows a screenshot of a Java IDE window titled "ButtonKeluar - UASPB0". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a tab bar with several tabs: "Buttons", "ButtonAbout", "ButtonBack", "ButtonBantuan", "ButtonKeluar", "ButtonMainlagi", "ButtonMulai", and "ButtonN". The "ButtonKeluar" tab is selected. Below the tab bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The main area of the IDE displays the source code for the ButtonKeluar class. The code starts with an import statement for greenfoot.* and a comment indicating it's for World, Actor, GreenfootImage, Greenfoot, and MouseInfo. It then has a class comment and Javadoc tags for author and version. The class is defined as public class ButtonKeluar extends Buttons. The act() method is implemented with two if statements: the first scales the image to 90% of its original size, and the second stops the game and sets the world to a new Beranda instance.

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

/**
 * Write a description of class ButtonKeluar here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ButtonKeluar extends Buttons

{
    /**
     * Act - do whatever the ButtonKeluar wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.mousePressed(this))
        {
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
            (int)Math.round(getImage().getHeight()*0.9));
        }
        if(Greenfoot.mousePressed(this))
        {
            Greenfoot.stop();
            Greenfoot.setWorld(new Beranda());
        }
    }
}
```

Kelas ButtonKeluar adalah subkelas dari Buttons dalam Greenfoot dan dirancang untuk merepresentasikan tombol "Keluar" dalam permainan.

Kelas ini bertindak sebagai representasi dari tombol "Keluar" yang dapat diinteraksi dalam permainan.

Merupakan subkelas dari Buttons, sehingga mewarisi sifat-sifat dasar dan perilaku dari kelas dasar tersebut.

Metode Act:

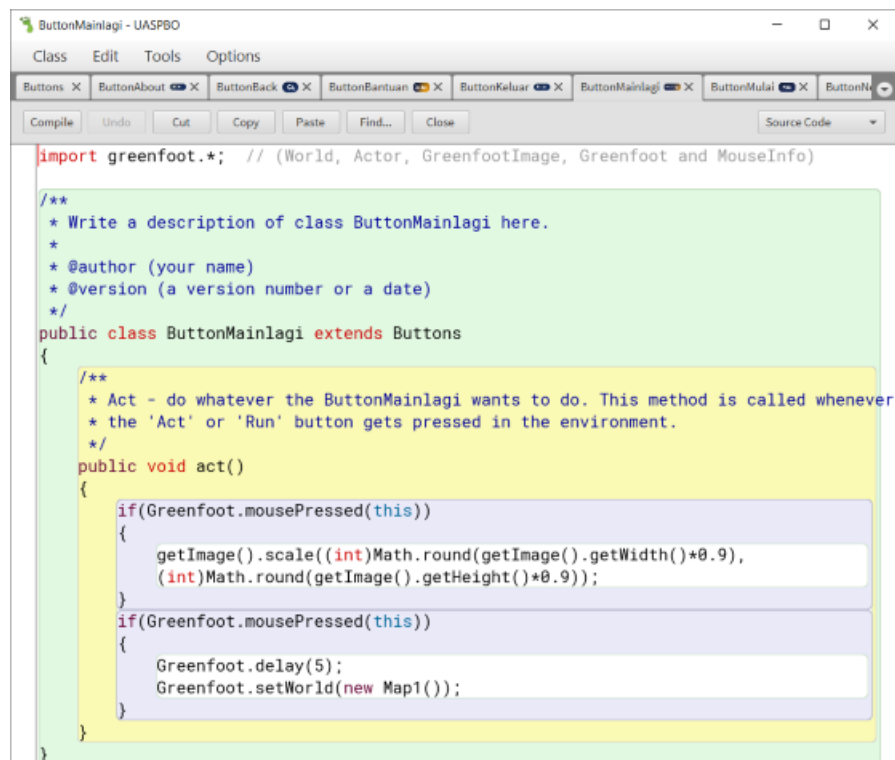
- Metode act() diimplementasikan untuk menanggapi aksi yang terjadi ketika tombol "Keluar" ditekan.

- Jika tombol "Keluar" ditekan, ukuran gambar tombol akan diperkecil sebesar 10% dari ukuran aslinya.

- Setelah itu, Greenfoot.stop() digunakan untuk menghentikan permainan, dan dunia akan diubah menjadi Beranda (layar beranda atau menu utama).

Dengan menggunakan pendekatan ini, kelas ButtonKeluar memberikan fungsionalitas khusus yang berkaitan dengan aksi saat tombol "Keluar" ditekan, sambil memanfaatkan kemampuan dan sifat dasar yang diwarisi dari kelas Buttons.

-Button MainLagi

The image shows a screenshot of a Java IDE window titled "ButtonMainlagi - UAS PBO". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a tab bar with several tabs: "Buttons", "ButtonAbout", "ButtonBack", "ButtonBantuan", "ButtonKeluar", "ButtonMainlagi", "ButtonMulai", and "ButtonN". The "ButtonMainlagi" tab is active. Below the tab bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A "Source Code" dropdown menu is also visible. The main area of the window displays the following Java code:

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

/**
 * Write a description of class ButtonMainlagi here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ButtonMainlagi extends Buttons
{
    /**
     * Act - do whatever the ButtonMainlagi wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.mousePressed(this))
        {
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
            (int)Math.round(getImage().getHeight()*0.9));
        }
        if(Greenfoot.mousePressed(this))
        {
            Greenfoot.delay(5);
            Greenfoot.setWorld(new Map1());
        }
    }
}
```

Kelas ButtonMainlagi adalah subkelas dari Buttons dalam Greenfoot dan dirancang untuk merepresentasikan tombol "Main Lagi" dalam permainan.

Kelas ini bertindak sebagai representasi dari tombol "Main Lagi" yang dapat diinteraksi dalam permainan.

Merupakan subkelas dari Buttons, sehingga mewarisi sifat-sifat dasar dan perilaku dari kelas dasar tersebut.

Metode Act:

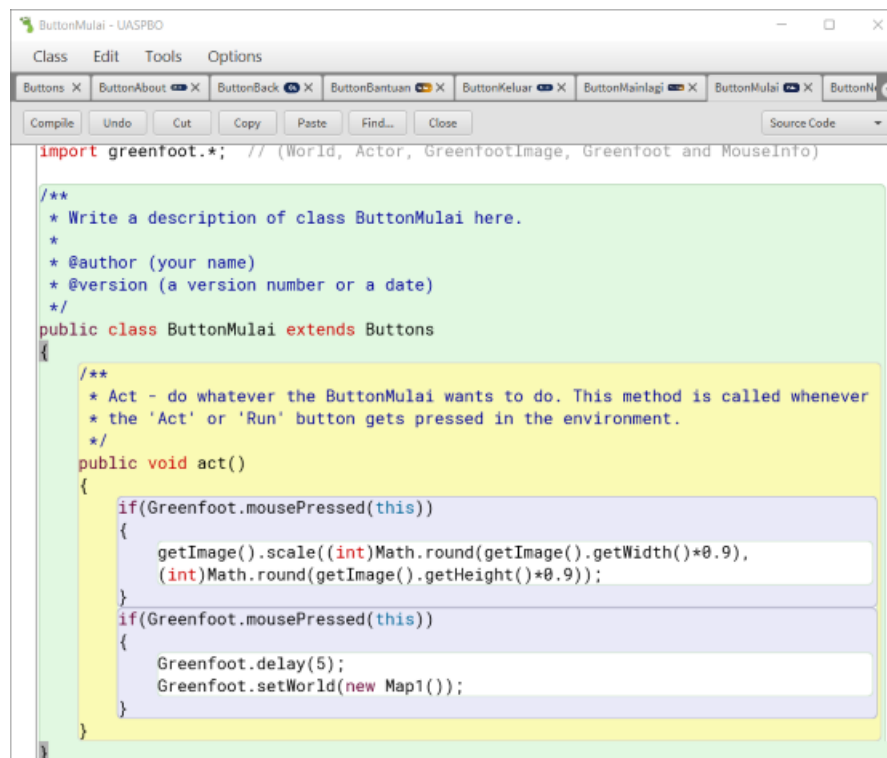
- Metode act() diimplementasikan untuk menanggapi aksi yang terjadi ketika tombol "Main Lagi" ditekan.

- Jika tombol "Main Lagi" ditekan, ukuran gambar tombol akan diperkecil sebesar 10% dari ukuran aslinya.

- Setelah itu, setelah penundaan sebentar menggunakan Greenfoot.delay(5), dunia akan diubah menjadi Map1 (memulai level atau tampilan permainan awal).

Dengan menggunakan pendekatan ini, kelas ButtonMainlagi memberikan fungsionalitas khusus yang berkaitan dengan aksi saat tombol "Main Lagi" ditekan, sambil memanfaatkan kemampuan dan sifat dasar yang diwarisi dari kelas Buttons.

-Button Mulai

A screenshot of a Java IDE window titled "ButtonMulai - UASPBIO". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a toolbar with buttons for "Buttons", "ButtonAbout", "ButtonBack", "ButtonBantuan", "ButtonKeluar", "ButtonMainlagi", "ButtonMulai", and "ButtonN". There is also a "Compile" button and a "Find..." button. The main area displays the source code for the ButtonMulai class. The code starts with an import statement for greenfoot.*. It then has a class comment and a class declaration: public class ButtonMulai extends Buttons. The act() method is defined, which contains two if statements. The first if statement checks for a mouse press and scales the image to 90% of its original size. The second if statement checks for a mouse press and delays the game for 5 units of time before setting the world to a new Map1().

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

/**
 * Write a description of class ButtonMulai here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ButtonMulai extends Buttons
{
    /**
     * Act - do whatever the ButtonMulai wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.mousePressed(this))
        {
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
                (int)Math.round(getImage().getHeight()*0.9));
        }
        if(Greenfoot.mousePressed(this))
        {
            Greenfoot.delay(5);
            Greenfoot.setWorld(new Map1());
        }
    }
}
```

Kelas ButtonMulai adalah subkelas dari Buttons dalam Greenfoot dan dirancang untuk merepresentasikan tombol "Mulai" dalam permainan.

Kelas ini bertindak sebagai representasi dari tombol "Mulai" yang dapat diinteraksi dalam permainan.

Merupakan subkelas dari Buttons, sehingga mewarisi sifat-sifat dasar dan perilaku dari kelas dasar tersebut.

Metode Act:

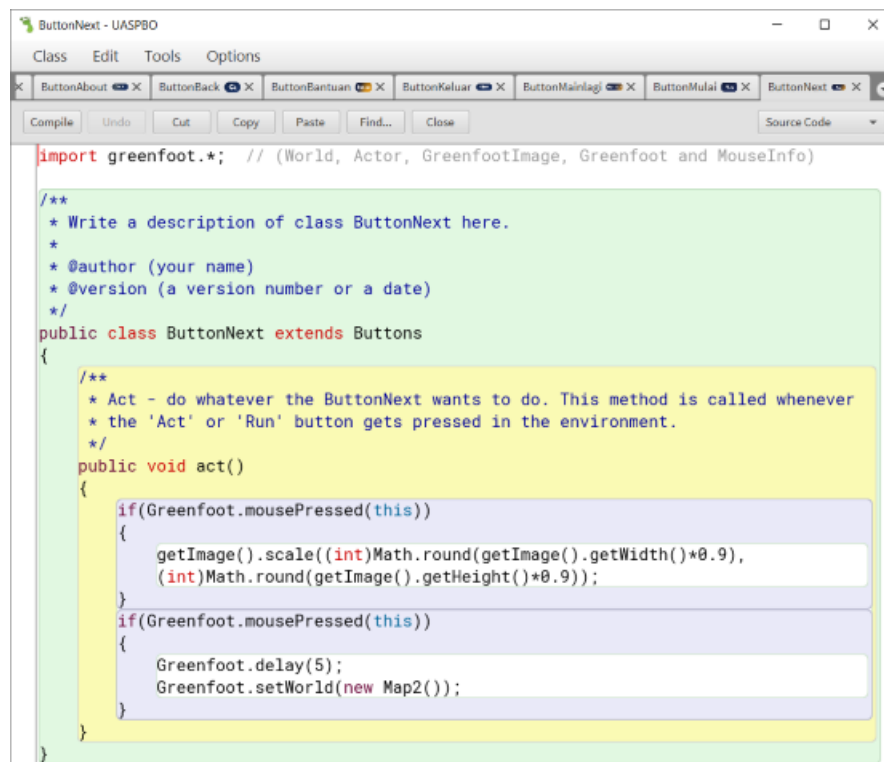
- Metode act() diimplementasikan untuk menanggapi aksi yang terjadi ketika tombol "Mulai" ditekan.

- Jika tombol "Mulai" ditekan, ukuran gambar tombol akan diperkecil sebesar 10% dari ukuran aslinya.

- Setelah itu, setelah penundaan sebentar menggunakan Greenfoot.delay(5), dunia akan diubah menjadi Map1 (memulai level atau tampilan permainan awal).

Dengan menggunakan pendekatan ini, kelas ButtonMulai memberikan fungsionalitas khusus yang berkaitan dengan aksi saat tombol "Mulai" ditekan, sambil memanfaatkan kemampuan dan sifat dasar yang diwarisi dari kelas Buttons.

-Button Next



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

/**
 * Write a description of class ButtonNext here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ButtonNext extends Buttons
{
    /**
     * Act - do whatever the ButtonNext wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.mousePressed(this))
        {
            getImage().scale((int)Math.round(getImage().getWidth()*0.9),
            (int)Math.round(getImage().getHeight()*0.9));
        }
        if(Greenfoot.mousePressed(this))
        {
            Greenfoot.delay(5);
            Greenfoot.setWorld(new Map2());
        }
    }
}
```

Kelas ButtonNext adalah subkelas dari Buttons dalam Greenfoot dan dirancang untuk merepresentasikan tombol "Next" dalam permainan.

Kelas ini bertindak sebagai representasi dari tombol "Next" yang dapat diinteraksi dalam permainan atau simulasi.

Merupakan subkelas dari Buttons, sehingga mewarisi sifat-sifat dasar dan perilaku dari kelas dasar tersebut.

Metode Act:

- Metode act() diimplementasikan untuk menanggapi aksi yang terjadi ketika tombol "Next" ditekan.

- Jika tombol "Next" ditekan, ukuran gambar tombol akan diperkecil sebesar 10% dari ukuran aslinya.

- Setelah itu, setelah penundaan sebentar menggunakan Greenfoot.delay(5), dunia akan diubah menjadi Map2 (memulai level atau tampilan permainan berikutnya).

Dengan menggunakan pendekatan ini, kelas ButtonNext memberikan fungsionalitas khusus yang berkaitan dengan aksi saat tombol "Next" ditekan, sambil memanfaatkan kemampuan dan sifat dasar yang diwarisi dari kelas Buttons.

- Class Karakter dan Class-Class Turunannya

-Class Karakter

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

/**
 * Write a description of class Karakter here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Karakter extends Actor
{
    /**
     * Act - do whatever the Karakter 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 Karakter adalah kelas dasar yang mewarisi dari Actor dalam Greenfoot.

Kelas ini bertindak sebagai kelas dasar untuk semua karakter atau aktor dalam permainan atau simulasi Greenfoot.

Merupakan subkelas dari Actor, sehingga dapat ditempatkan di dunia dan berinteraksi dengan elemen-elemen lainnya.

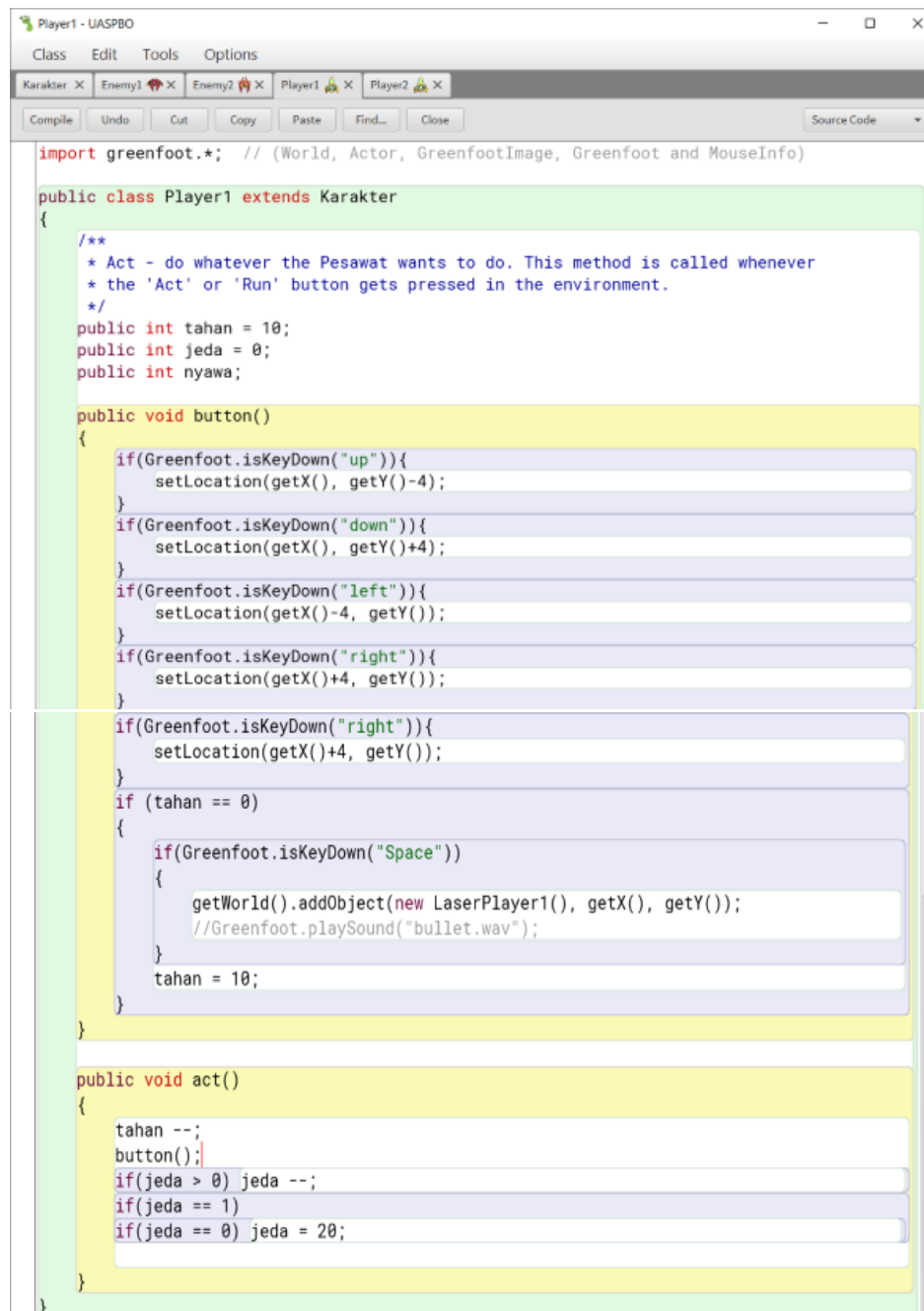
Metode Act:

-Metode act() tidak memiliki implementasi pada kelas ini.

-Pada kelas turunannya, metode act() dapat diperluas untuk menambahkan perilaku atau aksi yang diinginkan untuk karakter tertentu.

Dengan menggunakan pendekatan ini, kelas Karakter memberikan dasar untuk pembuatan karakter-karakter dalam permainan dan memungkinkan pengembang untuk menyesuaikan perilaku dan tampilan karakter sesuai dengan kebutuhan spesifik dalam permainan mereka.

-Class Player1



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

public class Player1 extends Karakter
{
    /**
     * Act - do whatever the Pesawat wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public int tahan = 10;
    public int jeda = 0;
    public int nyawa;

    public void button()
    {
        if(Greenfoot.isKeyDown("up")){
            setLocation(getX(), getY()-4);
        }
        if(Greenfoot.isKeyDown("down")){
            setLocation(getX(), getY()+4);
        }
        if(Greenfoot.isKeyDown("left")){
            setLocation(getX()-4, getY());
        }
        if(Greenfoot.isKeyDown("right")){
            setLocation(getX()+4, getY());
        }
        if(Greenfoot.isKeyDown("right")){
            setLocation(getX()+4, getY());
        }
        if (tahan == 0)
        {
            if(Greenfoot.isKeyDown("Space"))
            {
                getWorld().addObject(new LaserPlayer1(), getX(), getY());
                //Greenfoot.playSound("bullet.wav");
            }
            tahan = 10;
        }
    }

    public void act()
    {
        tahan --;
        button();
        if(jeda > 0) jeda --;
        if(jeda == 1)
        if(jeda == 0) jeda = 20;
    }
}
```

Kelas Player1 adalah turunan dari kelas Karakter dan dirancang untuk merepresentasikan karakter pemain pertama dalam permainan.

Variabel :

tahan: Digunakan untuk mengatur jeda atau cooldown antara penembakan.

jeda: Digunakan untuk mengatur jeda atau delay aksi tertentu.

nyawa: Variabel untuk menyimpan nilai jumlah nyawa karakter pemain.

Metode button():

Merupakan metode untuk menanggapi input dari pemain.

Jika tombol panah "atas" ditekan, pemain akan bergerak ke atas.

Jika tombol panah "bawah" ditekan, pemain akan bergerak ke bawah.

Jika tombol panah "kiri" ditekan, pemain akan bergerak ke kiri.

Jika tombol panah "kanan" ditekan, pemain akan bergerak ke kanan.

Jika tombol spasi ("Space") ditekan, pemain akan menembakkan proyektil (LaserPlayer1) ke arah yang dihadapi.

Metode act():

-Diimplementasikan dari kelas dasar dan digunakan untuk menjalankan aksi atau perilaku karakter.

-Mengurangi nilai tahan untuk mengatur jeda antara penembakan.

-Memanggil metode button() untuk menanggapi input pemain.

-Mengatur jeda (jeda) untuk suatu aksi tertentu (tidak dijelaskan dalam implementasi yang diberikan).

Dengan menggunakan pendekatan ini, kelas Player1 memberikan fungsionalitas untuk mengendalikan karakter pemain pertama, termasuk pergerakan dan penembakan.

-Class Player2



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

/**
 * Write a description of class Player2 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Player2 extends Karakter
{
    /**
     * Act - do whatever the Player2 wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public int tahan = 10;
    public int jeda = 0;
    public int nyawa;

    public void button()
    {
        if(Greenfoot.isKeyDown("up")){
            setLocation(getX(), getY()-4);
        }
        if(Greenfoot.isKeyDown("down")){
            setLocation(getX(), getY()+4);
        }
        if(Greenfoot.isKeyDown("left")){
            setLocation(getX()-4, getY());
        }
        if(Greenfoot.isKeyDown("right")){
            setLocation(getX()+4, getY());
        }
        if (tahan == 0)
        {
            if(Greenfoot.isKeyDown("Space"))
            {
                getWorld().addObject(new LaserPlayer2(), getX(), getY());
                //Greenfoot.playSound("bullet.wav");
            }
            tahan = 10;
        }
    }

    public void act()
    {
        tahan --;
        button();
        if(jeda > 0) jeda --;
        if(jeda == 1)
        if(jeda == 0) jeda = 20;
    }
}
```

Kelas Player2 adalah turunan dari kelas Karakter dan dirancang untuk merepresentasikan karakter pemain kedua dalam permainan.

Variabel Instans:

tahan: Digunakan untuk mengatur jeda atau cooldown antara penembakan.

jeda: Digunakan untuk mengatur jeda atau delay aksi tertentu.

nyawa: Variabel untuk menyimpan nilai jumlah nyawa karakter pemain.

Metode button():

Merupakan metode untuk menanggapi input dari pemain.

Jika tombol panah "atas" ditekan, pemain akan bergerak ke atas.

Jika tombol panah "bawah" ditekan, pemain akan bergerak ke bawah.

Jika tombol panah "kiri" ditekan, pemain akan bergerak ke kiri.

Jika tombol panah "kanan" ditekan, pemain akan bergerak ke kanan.

Jika tombol spasi ("Space") ditekan, pemain akan menembakkan proyektil (LaserPlayer2) ke arah yang dihadapi.

Metode act():

-Diimplementasikan dari kelas dasar dan digunakan untuk menjalankan aksi atau perilaku karakter.

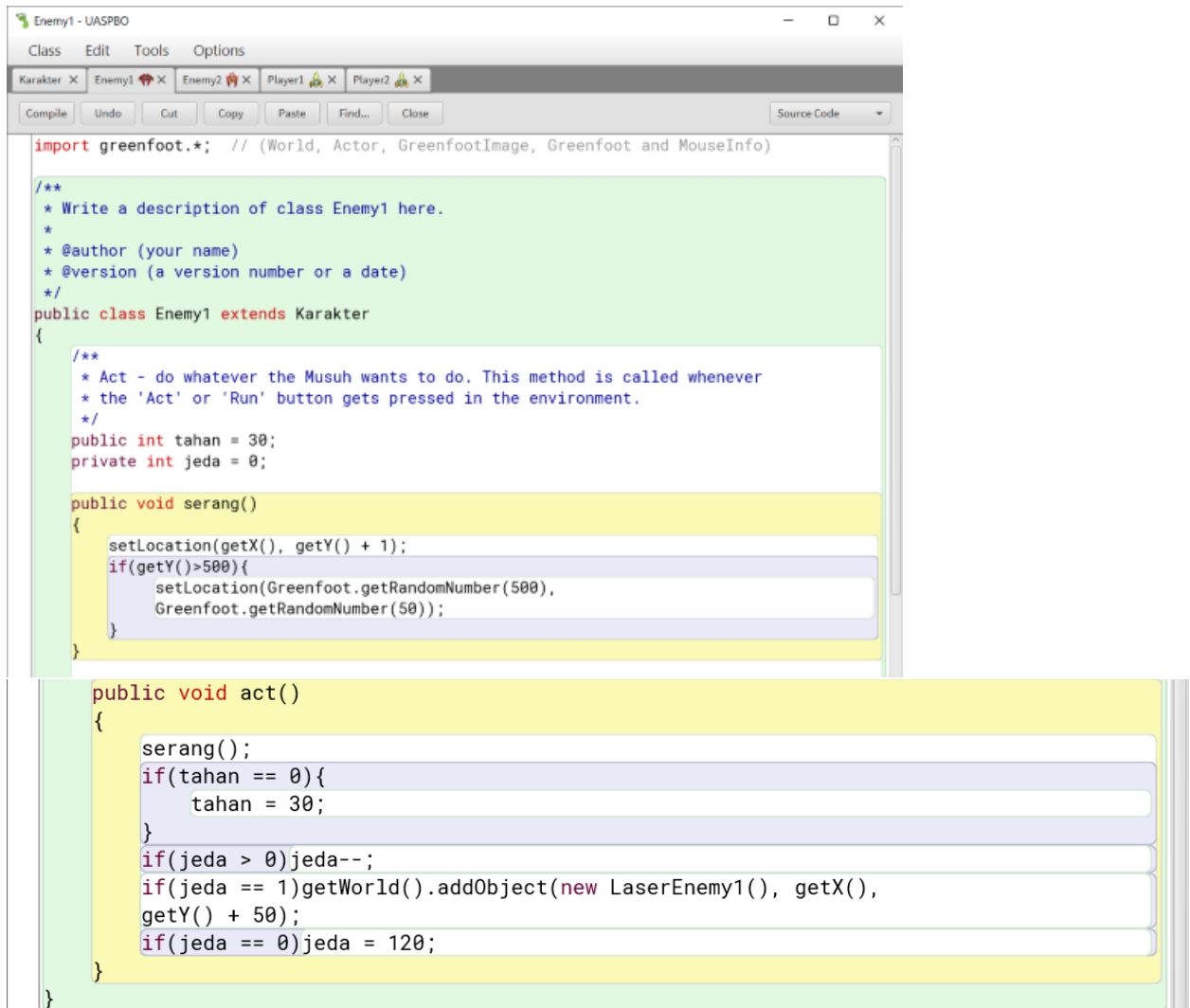
-Mengurangi nilai tahan untuk mengatur jeda antara penembakan.

-Memanggil metode button() untuk menanggapi input pemain.

-Mengatur jeda (jeda) untuk suatu aksi tertentu (tidak dijelaskan dalam implementasi yang diberikan).

Dengan menggunakan pendekatan ini, kelas Player2 memberikan fungsionalitas untuk mengendalikan karakter pemain kedua, termasuk pergerakan dan penembakan.

-Class Enemy1



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

/**
 * Write a description of class Enemy1 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Enemy1 extends Karakter
{
    /**
     * Act - do whatever the Musuh wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public int tahan = 30;
    private int jeda = 0;

    public void serang()
    {
        setLocation(getX(), getY() + 1);
        if(getY() > 500){
            setLocation(Greenfoot.getRandomNumber(500),
                Greenfoot.getRandomNumber(50));
        }
    }

    public void act()
    {
        serang();
        if(tahan == 0){
            tahan = 30;
        }
        if(jeda > 0)jeda--;
        if(jeda == 1)getWorld().addObject(new LaserEnemy1(), getX(),
            getY() + 50);
        if(jeda == 0)jeda = 120;
    }
}
```

Kelas Enemy1 adalah turunan dari kelas Karakter dan dirancang untuk merepresentasikan musuh pertama dalam permainan.

Variabel Instans:

tahan: Digunakan untuk mengatur jeda atau cooldown antara serangan musuh.

jeda: Digunakan untuk mengatur jeda atau delay antara serangan musuh.

Metode serang():

Merupakan metode untuk menanggapi serangan musuh.

Menggeser posisi musuh ke bawah (sumbu Y) sejauh 1 piksel.

Jika posisi musuh melebihi batas tertentu ($Y > 500$), musuh akan dipindahkan ke posisi acak di atas layar.

Metode act():

-Diimplementasikan dari kelas dasar dan digunakan untuk menjalankan aksi atau perilaku musuh.

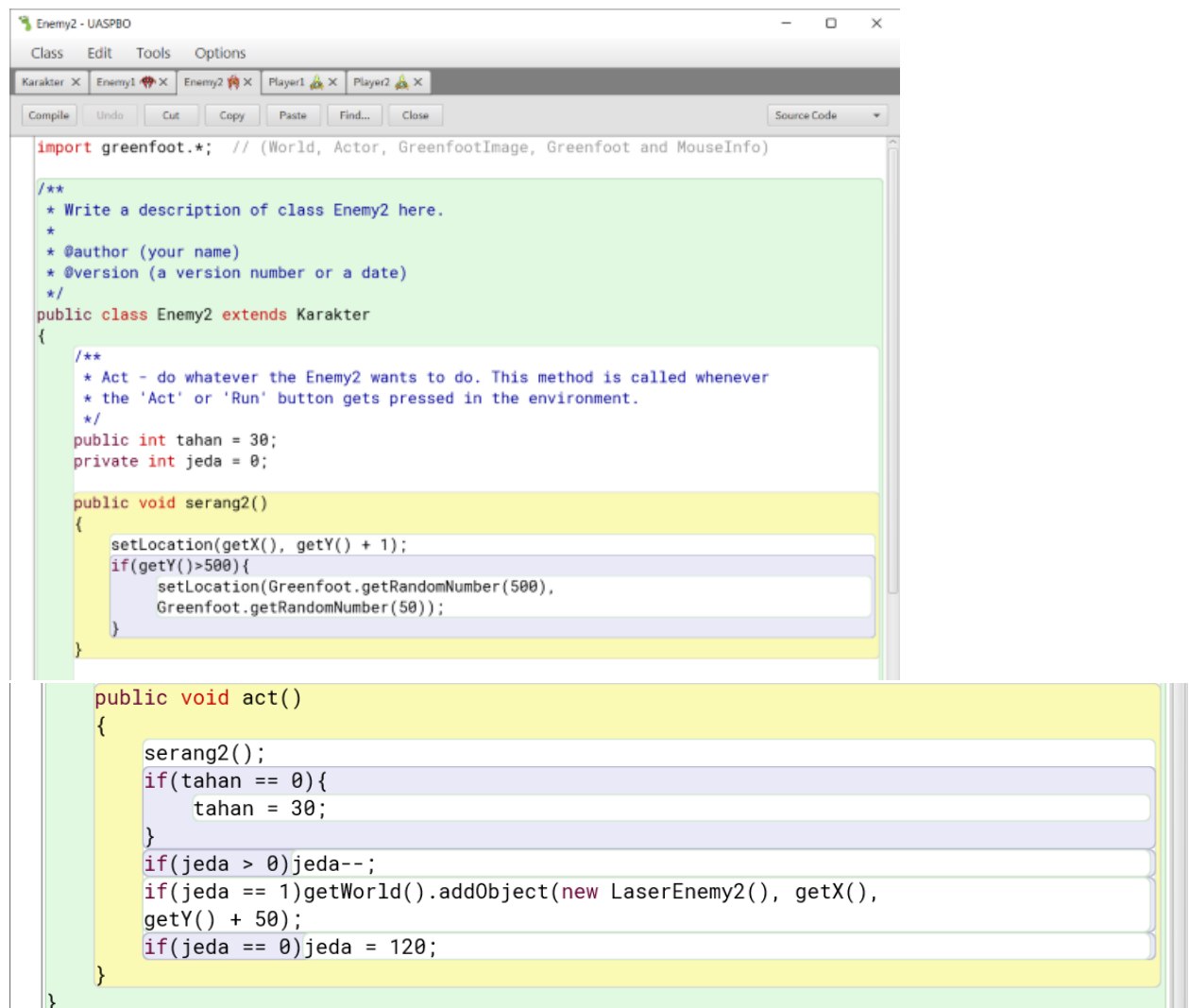
-Memanggil metode serang() untuk menjalankan serangan musuh.

-Mengurangi nilai tahan untuk mengatur jeda antara serangan musuh.

-Menambahkan proyektil musuh (LaserEnemy1) jika jeda mencapai nol.

Dengan menggunakan pendekatan ini, kelas Enemy1 memberikan fungsionalitas untuk mengatur perilaku musuh pertama, termasuk pergerakan dan serangan.

-Class Enemy2



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

/**
 * Write a description of class Enemy2 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Enemy2 extends Karakter
{
    /**
     * Act - do whatever the Enemy2 wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public int tahan = 30;
    private int jeda = 0;

    public void serang2()
    {
        setLocation(getX(), getY() + 1);
        if(getY() > 500){
            setLocation(Greenfoot.getRandomNumber(500),
                Greenfoot.getRandomNumber(50));
        }
    }

    public void act()
    {
        serang2();
        if(tahan == 0){
            tahan = 30;
        }
        if(jeda > 0)jeda--;
        if(jeda == 1)getWorld().addObject(new LaserEnemy2(), getX(),
            getY() + 50);
        if(jeda == 0)jeda = 120;
    }
}
```

Kelas Enemy2 adalah turunan dari kelas Karakter dan dirancang untuk merepresentasikan musuh kedua dalam permainan.

Variabel Instans:

tahan: Digunakan untuk mengatur jeda atau cooldown antara serangan musuh.

jeda: Digunakan untuk mengatur jeda atau delay antara serangan musuh.

Metode serang2():

Merupakan metode untuk menanggapi serangan musuh.

Menggeser posisi musuh ke bawah (sumbu Y) sejauh 1 piksel.

Jika posisi musuh melebihi batas tertentu ($Y > 500$), musuh akan dipindahkan ke posisi acak di atas layar.

Metode act():

-Diimplementasikan dari kelas dasar dan digunakan untuk menjalankan aksi atau perilaku musuh.

-Memanggil metode serang2() untuk menjalankan serangan musuh.

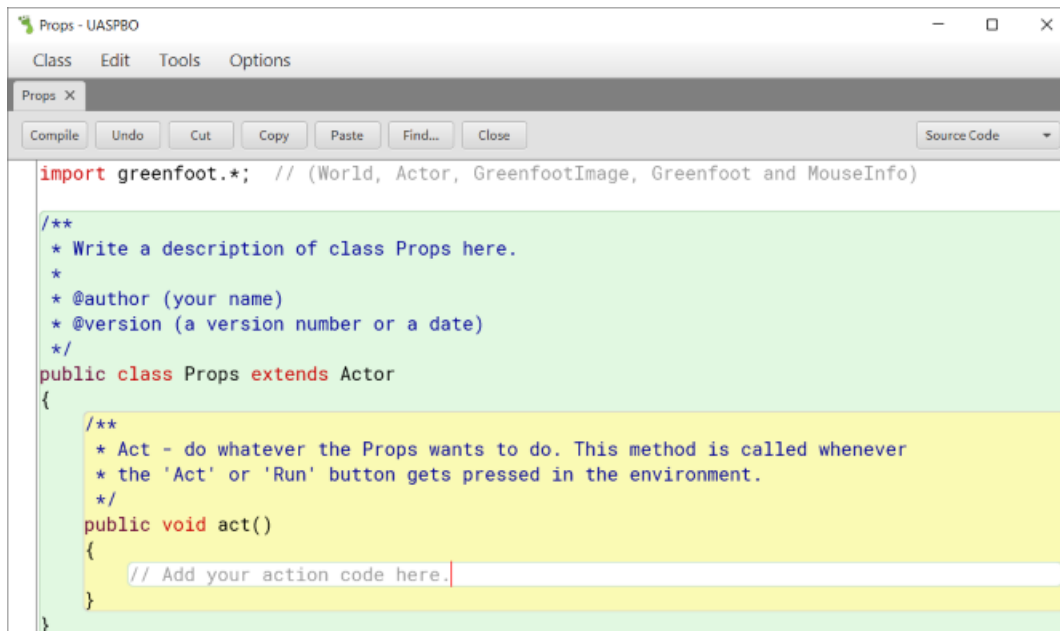
-Mengurangi nilai tahan untuk mengatur jeda antara serangan musuh.

-Menambahkan proyektil musuh (LaserEnemy2) jika jeda mencapai nol.

Dengan menggunakan pendekatan ini, kelas Enemy2 memberikan fungsionalitas untuk mengatur perilaku musuh kedua, termasuk pergerakan dan serangan.

- Class Props dan Class-Class Turunannya

-Class Props



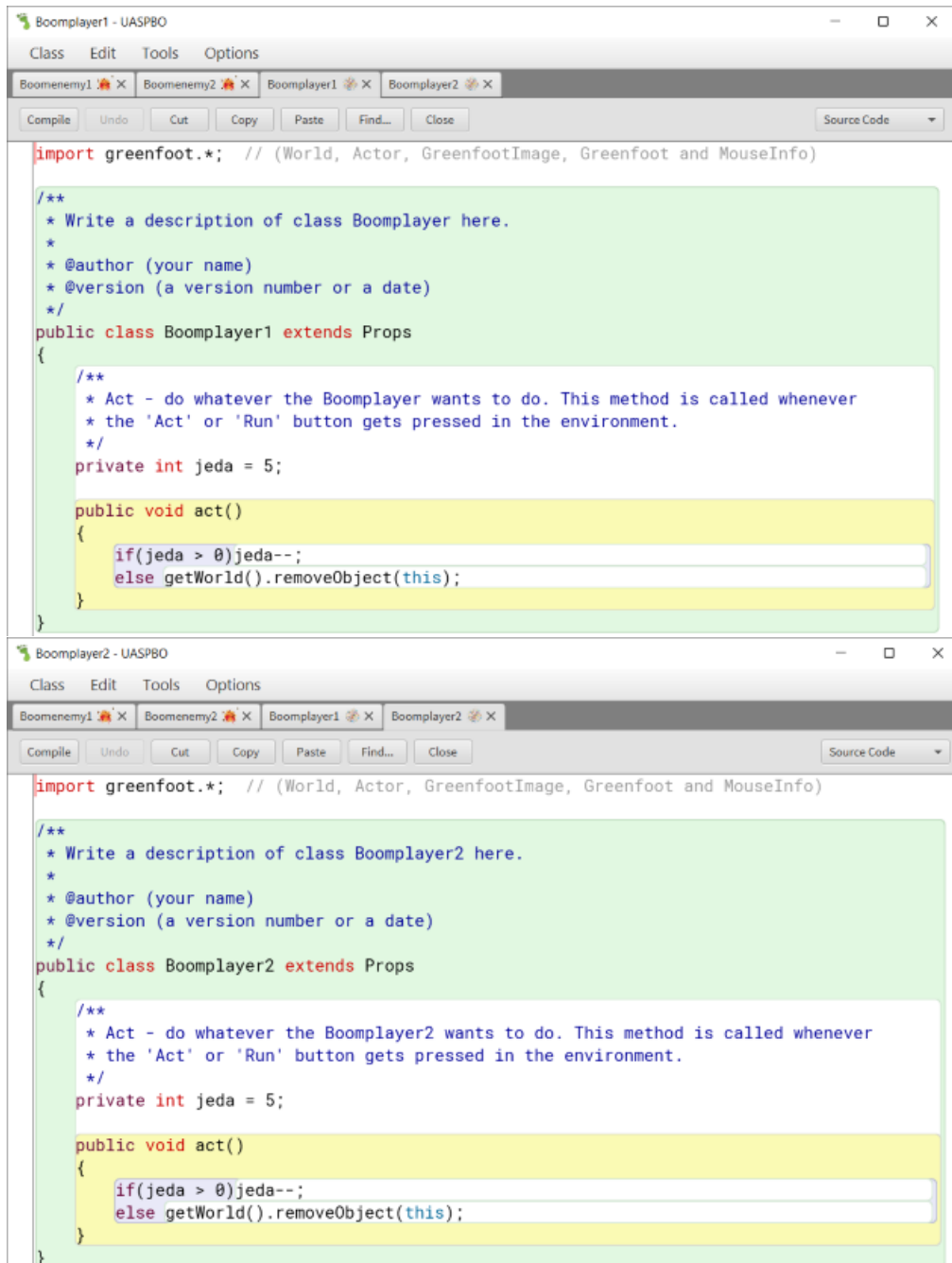
Kelas Props adalah turunan dari kelas Actor dan dirancang untuk merepresentasikan objek atau elemen lain dalam lingkungan permainan yang bukan karakter pemain atau musuh.

Metode act():

- Diimplementasikan dari kelas dasar Actor.
- Digunakan untuk menjalankan aksi atau perilaku objek atau elemen props.
- Metode ini tidak diisi dengan kode apapun.

Kelas Props berfungsi sebagai dasar untuk objek atau elemen tambahan dalam permainan yang bukan karakter utama atau musuh. Objek-objek ini bisa termasuk hiasan, platform, atau elemen lingkungan lainnya yang tidak terlibat dalam interaksi langsung dengan pemain atau musuh.

-Class BoomPlayer1 dan BoomPlayer2



Kelas Boomplayer1 dan Boomplayer2 adalah turunan dari kelas Props dan dirancang untuk merepresentasikan efek atau objek yang terkait dengan pemain pertama (Player1) dan pemain kedua (Player2) dalam permainan.

Variabel Instans:

jeda: Digunakan untuk mengatur jeda atau durasi objek Boomplayer1 dan Boomplayer2 sebelum dihapus dari dunia.

Metode act():

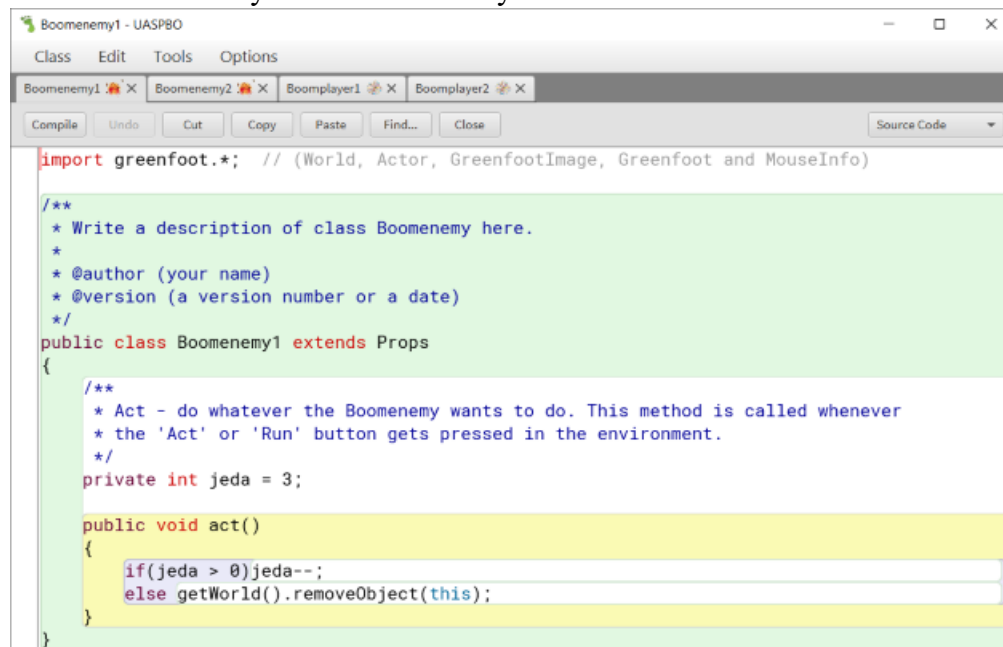
-Diimplementasikan dari kelas dasar Props.

-Mengurangi nilai jeda untuk mengatur durasi objek Boomplayer1 dan Boomplayer2.

-Jika jeda mencapai nol, objek Boomplayer1 dan Boomplayer2 dihapus dari dunia menggunakan `getWorld().removeObject(this)`.

Kedua Kelas ini mewakili efek atau objek sementara yang terkait dengan pemain pertama dan kedua dalam permainan, dan objek ini akan muncul untuk beberapa waktu sebelum menghilang.

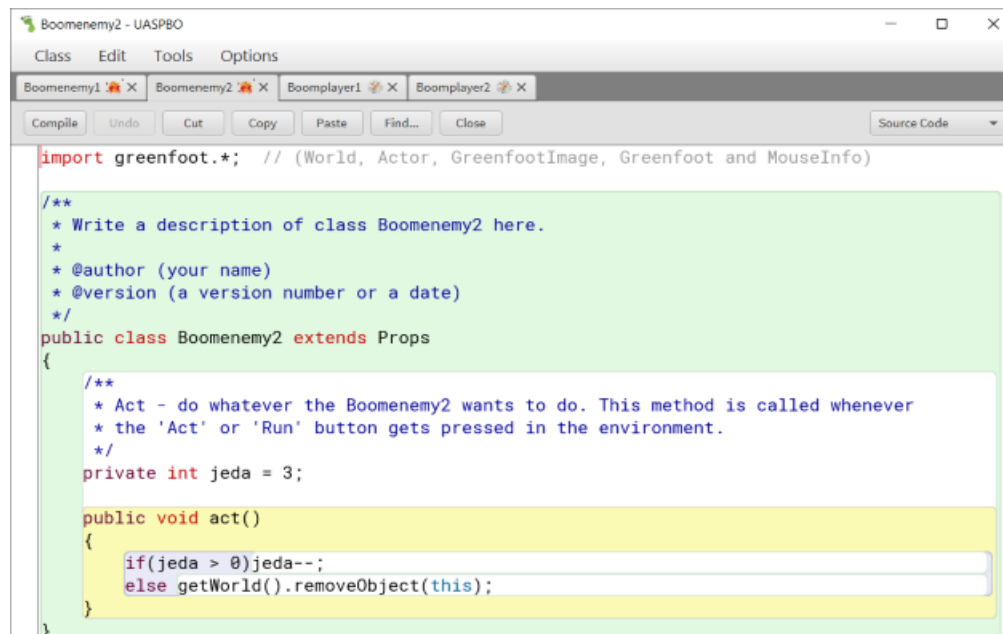
-Class BoomEnemy1 dan BoomEnemy2



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

/**
 * Write a description of class Boomenemy here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Boomenemy1 extends Props
{
    /**
     * Act - do whatever the Boomenemy wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    private int jeda = 3;

    public void act()
    {
        if(jeda > 0)jeda--;
        else getWorld().removeObject(this);
    }
}
```



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

/**
 * Write a description of class Boomenemy2 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Boomenemy2 extends Props
{
    /**
     * Act - do whatever the Boomenemy2 wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    private int jeda = 3;

    public void act()
    {
        if(jeda > 0)jeda--;
        else getWorld().removeObject(this);
    }
}
```

Kelas Boomenemy1 dan Boomenemy2 adalah turunan dari kelas Props dan dirancang untuk merepresentasikan efek atau objek yang terkait dengan musuh pertama (Enemy1) dan musuh kedua (Enemy2) dalam permainan.

Variabel Instans:

jeda: Digunakan untuk mengatur jeda atau durasi objek Boomenemy1 dan Boomenemy2 sebelum dihapus dari dunia.

Metode act():

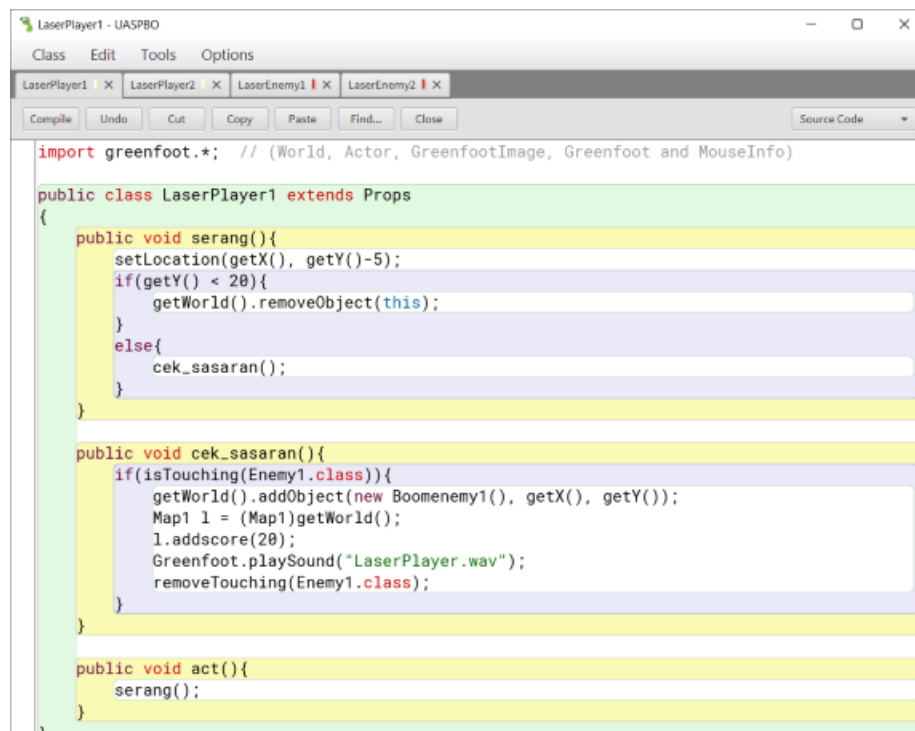
- Diimplementasikan dari kelas dasar Props.

- Mengurangi nilai jeda untuk mengatur durasi objek Boomenemy1 dan Boomenemy2.

- Jika jeda mencapai nol, objek Boomenemy1 dan Boomenemy2 dihapus dari dunia menggunakan `getWorld().removeObject(this)`.

Kedua kelas ini mewakili efek atau objek sementara yang terkait dengan musuh pertama dan musuh kedua dalam permainan, dan objek ini akan muncul untuk beberapa waktu sebelum menghilang.

- Class LaserPlayer1 dan LaserPlayer2



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

public class LaserPlayer1 extends Props
{
    public void serang(){
        setLocation(getX(), getY()-5);
        if(getY() < 20){
            getWorld().removeObject(this);
        }
        else{
            cek_sasaran();
        }
    }

    public void cek_sasaran(){
        if(isTouching(Enemy1.class)){
            getWorld().addObject(new Boomenemy1(), getX(), getY());
            Map1 l = (Map1)getWorld();
            l.addscore(20);
            Greenfoot.playSound("LaserPlayer.wav");
            removeTouching(Enemy1.class);
        }
    }

    public void act(){
        serang();
    }
}
```

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

public class LaserPlayer2 extends Props
{
    public void serang2(){
        setLocation(getX(), getY()-5);
        if(getY() < 20){
            getWorld().removeObject(this);
        }
        else{
            cek_sasaran2();
        }
    }

    public void cek_sasaran2(){
        if(isTouching(Enemy2.class)){
            getWorld().addObject(new Boomenemy2(), getX(), getY());
            Map2 t = (Map2)getWorld();
            t.addscore(20);
            Greenfoot.playSound("LaserPlayer.wav");
            removeTouching(Enemy2.class);
        }
    }

    public void act(){
        serang2();
    }
}
```

Kelas LaserPlayer1 dan LaserPlayer2 adalah turunan dari kelas Props dan dirancang untuk merepresentasikan proyektil atau serangan yang dilakukan oleh pemain pertama (Player1) dan pemain kedua (Player2) dalam permainan.

Metode serang() dan Metode serang2():

- Menggerakkan objek LaserPlayer1 dan LaserPlayer2 ke atas (mengurangi nilai Y).
- Jika posisi Y objek LaserPlayer1 dan LaserPlayer2 kurang dari 20, objek dihapus dari dunia.
- Jika tidak, memanggil metode cek_sasaran() untuk memeriksa apakah objek menyentuh musuh.

Metode cek_sasaran() dan Method cek_sasaran2():

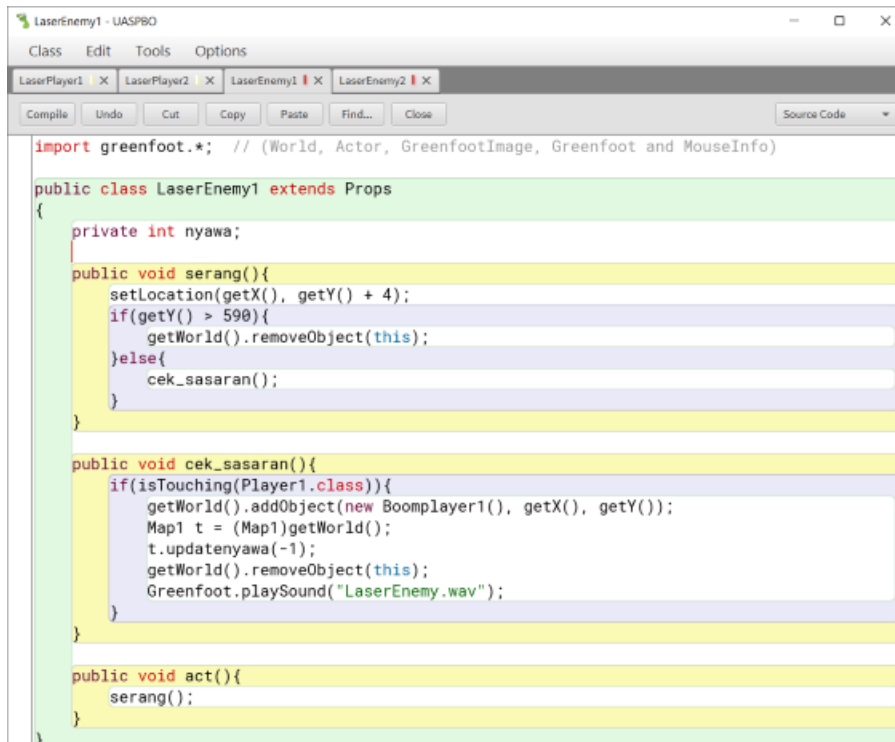
- Memeriksa apakah objek LaserPlayer1 dan LaserPlayer2 menyentuh objek dari kelas Enemy1 dan Enemy2.
 - Jika ya, menambahkan objek Boomenemy1 dan Boomenemy2 ke dunia di posisi objek LaserPlayer1 dan LaserPlayer2.
 - Mendapatkan referensi ke dunia saat ini (Map1) dan (Map2) dan memanggil metode addscore(20) untuk menambah skor.
 - Memutar suara efek tembakan menggunakan Greenfoot.playSound("LaserPlayer.wav").
- Menghapus objek musuh yang disentuh.

Metode act():

Memanggil metode serang() dan metode serang2().

Kedua kelas ini mewakili logika serangan dari pemain pertama dan pemain kedua dalam permainan. Dan menyesuaikan nilai-nilai seperti pergerakan, pengecekan sasaran, skor yang ditambahkan, dan efek suara sesuai dengan permainan.

-Class LaserEnemy1

The image shows a screenshot of a Java IDE window titled 'LaserEnemy1 - UASPB0'. The window has a menu bar with 'Class', 'Edit', 'Tools', and 'Options'. Below the menu bar are tabs for 'LaserPlayer1', 'LaserPlayer2', 'LaserEnemy1', and 'LaserEnemy2'. The 'LaserEnemy1' tab is active, showing the source code. The code is as follows:

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

public class LaserEnemy1 extends Props
{
    private int nyawa;

    public void serang(){
        setLocation(getX(), getY() + 4);
        if(getY() > 590){
            getWorld().removeObject(this);
        }else{
            cek_sasaran();
        }
    }

    public void cek_sasaran(){
        if(isTouching(Player1.class)){
            getWorld().addObject(new Boomplayer1(), getX(), getY());
            Map1 t = (Map1)getWorld();
            t.updatenyawa(-1);
            getWorld().removeObject(this);
            Greenfoot.playSound("LaserEnemy.wav");
        }
    }

    public void act(){
        serang();
    }
}
```

Kelas LaserEnemy1 adalah turunan dari kelas Props dan dirancang untuk merepresentasikan proyektil atau serangan yang dilakukan oleh musuh pertama (Enemy1) dalam permainan.

Variabel nyawa:

Menyimpan jumlah nyawa.

Metode serang():

-Menggerakkan objek LaserEnemy1 ke bawah (menambah nilai Y).

-Jika posisi Y objek LaserEnemy1 lebih dari 590, objek dihapus dari dunia.

-Jika tidak, memanggil metode cek_sasaran() untuk memeriksa apakah objek menyentuh pemain.

Metode cek_sasaran():

- Memeriksa apakah objek LaserEnemy1 menyentuh objek dari kelas Player1.
- Jika ya, menambahkan objek Boomplayer1 ke dunia di posisi objek LaserEnemy1.
- Mendapatkan referensi ke dunia saat ini (Map1) dan memanggil metode updatenyawa(-1) untuk mengurangi nyawa pemain.
- Menghapus objek LaserEnemy1 dari dunia.
- Memutar suara efek tembakan menggunakan Greenfoot.playSound("LaserEnemy.wav").

Metode act():

- Memanggil metode serang().

Kelas ini mewakili logika serangan dari musuh pertama dalam permainan. Menyesuaikan nilai-nilai seperti pergerakan, pengecekan sasaran, pengurangan nyawa, dan efek suara sesuai dengan kebutuhan desain permainan.

-Class LaserEnemy2

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

public class LaserEnemy2 extends Props
{
    private int nyawa;

    public void serang2(){
        setLocation(getX(), getY() + 4);
        if(getY() > 590){
            getWorld().removeObject(this);
        }else{
            cek_sasaran2();
        }
    }

    public void cek_sasaran2(){
        if(isTouching(Player2.class)){
            getWorld().addObject(new Boomplayer2(), getX(), getY());
            Map2 l = (Map2)getWorld();
            l.updatenyawa(-1);
            getWorld().removeObject(this);
            Greenfoot.playSound("LaserEnemy.wav");
        }
    }

    public void act(){
        serang2();
    }
}
```

Kelas LaserEnemy2 adalah turunan dari kelas Props dan dirancang untuk merepresentasikan proyektil atau serangan yang dilakukan oleh musuh kedua (Enemy2) dalam permainan.

Kelas LaserEnemy2:

Variabel nyawa:

Menyimpan jumlah nyawa.

Metode serang2():

- Menggerakkan objek LaserEnemy2 ke bawah (menambah nilai Y).
- Jika posisi Y objek LaserEnemy2 lebih dari 590, objek dihapus dari dunia.
- Jika tidak, memanggil metode cek_sasaran2() untuk memeriksa apakah objek menyentuh pemain.

Metode cek_sasaran2():

- Memeriksa apakah objek LaserEnemy2 menyentuh objek dari kelas Player2.
- Jika ya, menambahkan objek Boomplayer2 ke dunia di posisi objek LaserEnemy2.
- Mendapatkan referensi ke dunia saat ini (Map2) dan memanggil metode updatenyawa(-1) untuk mengurangi nyawa pemain.
- Menghapus objek LaserEnemy2 dari dunia.
- Memutar suara efek tembakan menggunakan Greenfoot.playSound("LaserEnemy.wav").

Metode act():

- Memanggil metode serang2().

Kelas ini mewakili logika serangan dari musuh kedua dalam permainan. Pemrogram dapat menyesuaikan nilai-nilai seperti pergerakan, pengecekan sasaran, pengurangan nyawa, dan efek suara sesuai dengan kebutuhan desain permainan.

4. Penjelasan Tentang Game Shooter (Space Galaxy) Yang Dibuat.

- Tujuan Game : Tujuan pemain adalah bertahan hidup selama mungkin dan mengumpulkan skor sebanyak mungkin dengan cara menghindari musuh dan menembak musuh-musuh.
- Level : Terdapat dua tingkat (level) permainan, direpresentasikan oleh kelas Map1 dan Map2. Setiap level mungkin memiliki tingkat kesulitan yang berbeda.
- Pemain : Terdapat dua karakter pemain, direpresentasikan oleh kelas Player1 dan Player2. Masing-masing pemain dapat bergerak dan menembak.
- Musuh : Terdapat dua tipe musuh, direpresentasikan oleh kelas Enemy1 dan Enemy2. Musuh muncul secara acak di layar dan menyerang pemain.
- Bantuan : Pemain mengendalikan karakter menggunakan tombol-tombol navigasi seperti "up," "down," "left," dan "right". Dan Tombol "Space" digunakan untuk menembak.
- Interaksi : Pemain berinteraksi dengan tombol-tombol menu seperti "Mulai," "Bantuan," dan "Tentang". Pemain juga berinteraksi dengan musuh dan objek-objek lain di layar.
- Efek atau Props : Terdapat efek-efek seperti ledakan (Boomplayer1, Boomplayer2, Boomenemy1, Boomenemy2) dan serangan laser (LaserPlayer1, LaserPlayer2, LaserEnemy1, LaserEnemy2).
- Skor dan Nyawa : Pemain memiliki skor yang meningkat setiap kali mereka menyerang atau mengalahkan musuh. Nyawa pemain berkurang ketika terkena serangan musuh.
- Menu : Terdapat menu awal (Beranda) dan menu selama permainan (Menu). Menu dapat membawa pemain ke berbagai bagian permainan seperti tingkat, bantuan, atau informasi tentang permainan. Pemain juga dapat beralih antara layar-layar permainan menggunakan tombol-tombol menu seperti "Mulai," "Bantuan," dan "Tentang".

Deskripsi Tugas:

1. Kelas dan Objek: Penerapan kelas dan objek yang jelas dan sesuai dengan konsep PBO/OOP.

-Terdapat banyak kelas, seperti Menu, MenuAwal, MenuAbout, MenuBantuan, PlayAgain, Win, Buttons, ButtonAbout, ButtonBack, ButtonBantuan, ButtonKeluar, ButtonMainlagi, ButtonMulai, ButtonNext, Karakter, Player1, Player2, Enemy1, Enemy2, Props, Boomplayer1, Boomplayer2, Boomenemy1, Boomenemy2, LaserPlayer1, LaserPlayer2, LaserEnemy1, dan LaserEnemy2.

-Objek-objek dari kelas-kelas ini dibuat dan digunakan di dalam program.

2. Pewarisan (Inheritance): Penggunaan pewarisan untuk membangun hubungan antar kelas yang logis dan efisien.

-Terdapat pewarisan pada beberapa kelas, seperti MenuAbout, MenuAwal, dan MenuBantuan yang merupakan pewaris dari kelas Menu. Begitu juga dengan kelas PlayAgain, Win, dan NextLevel yang merupakan pewaris dari kelas World.

3. Polimorfisme: Pemanfaatan polimorfisme untuk menciptakan variasi dalam perilaku objek.

-Polimorfisme terlihat dalam penggunaan metode act() pada kelas-kelas seperti ButtonAbout, ButtonBack, ButtonBantuan, ButtonKeluar, ButtonMainlagi, ButtonMulai, ButtonNext, Player1, Player2, Enemy1, Enemy2, Props, Boomplayer1, Boomplayer2, Boomenemy1, Boomenemy2, LaserPlayer1, LaserPlayer2, LaserEnemy1, dan LaserEnemy2. Metode act() dapat memiliki implementasi yang berbeda di setiap kelas tersebut.

4. Enkapsulasi: Penggunaan enkapsulasi untuk melindungi data dan memastikan akses yang aman.

-Penggunaan variabel dan metode dalam suatu kelas dapat dianggap sebagai langkah-langkah enkapsulasi, di mana detail implementasi di dalam kelas tidak diungkapkan secara langsung ke kelas lain.

5. Interaksi Antar Objek: Implementasi interaksi antar objek yang dinamis dan menarik.

-Interaksi antar objek terjadi ketika objek-objek seperti Player1, Player2, Enemy1, Enemy2, LaserPlayer1, LaserPlayer2, LaserEnemy1, dan LaserEnemy2 saling berinteraksi melalui metode-metode seperti serang(), cek_sasaran(), dan lainnya.

6. Overriding dan Overloading: Penggunaan overriding untuk mengganti perilaku metode dari superclass dan overloading untuk membuat metode dengan parameter yang berbebea.

-Terdapat overriding dalam metode act() pada banyak kelas yang merupakan turunan dari Actor. Sebagai contoh, ButtonAbout, ButtonBack, ButtonBantuan, ButtonKeluar, ButtonMainlagi, ButtonMulai, ButtonNext, Player1, Player2, Enemy1, Enemy2, Props, Boomplayer1, Boomplayer2, Boomenemy1, Boomenemy2, LaserPlayer1, LaserPlayer2, LaserEnemy1, dan LaserEnemy2 semuanya memiliki metode act() yang diambil dari kelas induk (Actor) dan di-overriding.

-Overloading terjadi Pada kelas Buttons, kita melihat overloading pada metode act. Overriding terjadi ketika sebuah kelas anak (subkelas) menyediakan implementasi yang berbeda untuk metode yang sudah didefinisikan di kelas induknya (superclass). Dalam Greenfoot, metode act sering kali di-overriding di kelas-kelas Actor untuk memberikan perilaku yang khusus untuk objek tersebut.