

**LAPORAN UAS PRAKTIKUM PEMROGRAMAN
BERIORENTASIKAN OBJEK**

“GAME MOBIL”



Disusun Oleh:

Muhammad Rizqi 2211102441109

Rusdiyansyah 2211102441090

Muhammad Farhan 2211102441121

Kelas B

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

Daftar Isi

BAB I PENDAHULUAN.....	3
1.1 Latar Belakang	3
1.2 Tujuan.....	3
1.3 Manfaat.....	3
BAB II PEMBAHASAN	4
2.1 MyWorld.....	4
2.2 YouWin.....	6
2.3 GameOver	7
2.4 Karakter.....	7
2.5 Mobil1	8
2.6 Mobil2.....	9
2.7 Mobil4.....	10
2.8 Props	11
2.9 Blast	12
2.10 Counter	13
2.11 Peluru	14
BAB III PENUTUP	15
3.1 Kesimpulan	15
3.2 Saran	15

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengembangan permainan menjadi relevan karena menggabungkan konsep PBO dengan kreativitas dalam menciptakan suatu produk yang interaktif. Game ini memberikan kesempatan mengembangkan konsep-konsep PBO. Dengan mengeksplorasi pengembangan game mobil ini memberikan pengalaman praktis yang bisa dinikmati.

1.2 Tujuan

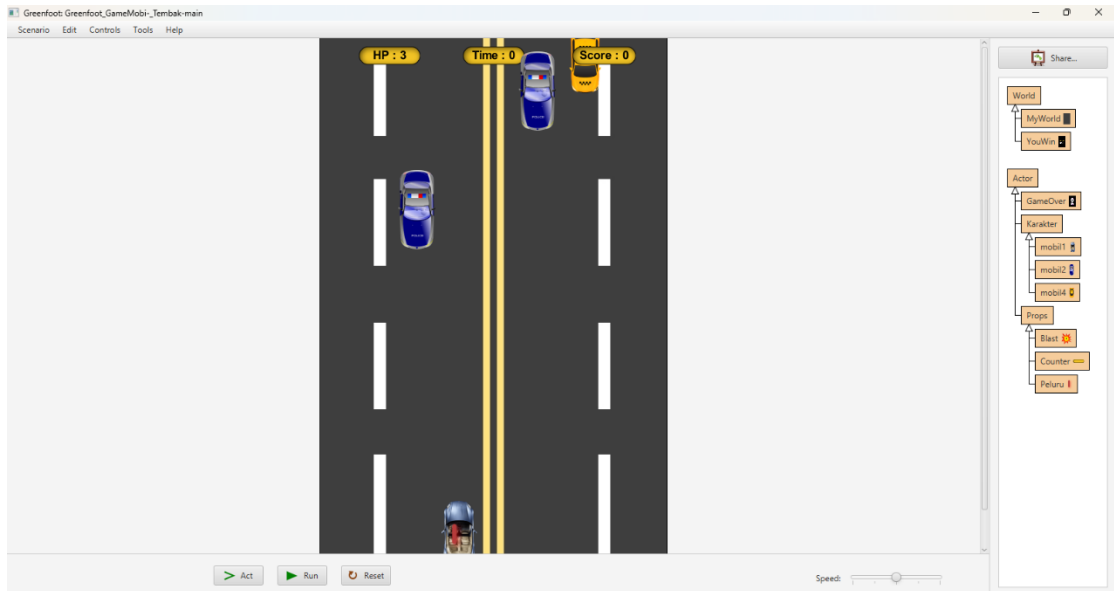
Tujuan dari membuat laporan ini ialah menerapkan prinsip-prinsip PBO dalam pengembangan game, meningkatkan keterampilan pemrograman, memahami elemen-elemen kunci dalam pengembangan game dan menunjukkan kreativitas dalam desain.

1.3 Manfaat

Manfaatnya untuk pengembangan dalam pembuatan game memakai Greenfoot. Penerapan dari hasil pembelajaran yang diberikan dosen pengampu. Mengembangkan kreatifitas dalam pembuatan game. Dan meningkatkan pengalaman pengembangan dari segi tim.

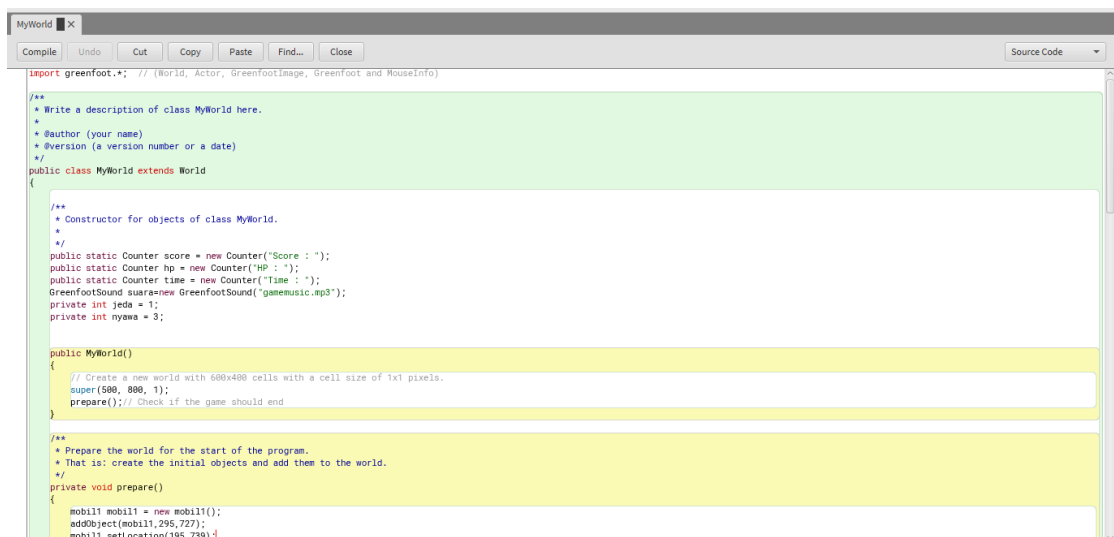
BAB II

PEMBAHASAN



Berikut ini adalah beberapa class dan penjelasannya:

2.1 MyWorld



```

mobil2 mobil2 = new mobil2();
addObject(mobil2, 213, 112);
mobil14 mobil14 = new mobil14();
addObject(mobil14, 43, 75);
mobil12.setLocation(118, 137);
mobil12.setLocation(258, 384);
mobil12.setLocation(269, 255);
mobil12.setLocation(220, 114);
mobil14.setLocation(49, 93);
mobil12.setLocation(233, 115);
removeObject(mobil12);
mobil14.setLocation(49, 89);

addObject(mobil14, 280, 124);
addObject(mobil14, 355, 74);
mobil14.setLocation(364, 73);
mobil14.setLocation(382, 31);
mobil11.setLocation(195, 748);
mobil11.setLocation(196, 717);
Peluru peluru = new Peluru();
addObject(peluru, 196, 717);
addObject(score, 57, 26);
score.setLocation(488, 25);
score.setLocation(489, 25);
score.setValue(0);
mobil14 mobil144 = new mobil14();
addObject(mobil144, 382, 223);
mobil144.setLocation(244, 32);
mobil144.setLocation(254, 25);
removeObject(mobil144);
addObject(hp, 180, 25);
hp.setValue(3);
addObject(time, 258, 25);
time.setValue(0);

mobil2 mobil22 = new mobil2();
addObject(mobil22, 313, 76);
mobil12 mobil123 = new mobil12();
addObject(mobil123, 139, 247);
}

public void act()
{
    countTime(); // Call countTime() to increment time continuously
    checkGameEnd(); // Check if the game should end// Call countTime() to increment time continuously
    // ... existing code ...
}

private void countTime()
{
    if (jeda > 0)
    {
        jeda--;
        return;
    }

    time.add(1); // Increment the time
    // Update the displayed time

    jeda = 3; // Set a delay to control the interval at which time increases.
    // You can adjust the interval and other parameters as needed.
    if (time.getValue() % 100 == 0)
    {
        // Add any additional actions you want to perform at this interval.
    }
}

private void showTime()
{
    showText("Time: " + time.getValue(), 200, 25);
}

private void checkGameEnd()
{
    if (time.getValue() >= 300 || score.getValue() >= 100)
    {
        // Game ends. player wins

        YouWin youWin = new YouWin();
        Greenfoot.setWorld(youWin);
        youWin.playWinSound();
        Greenfoot.stop();
    }
}

public void started()
{
    suara.playLoop();
}

```

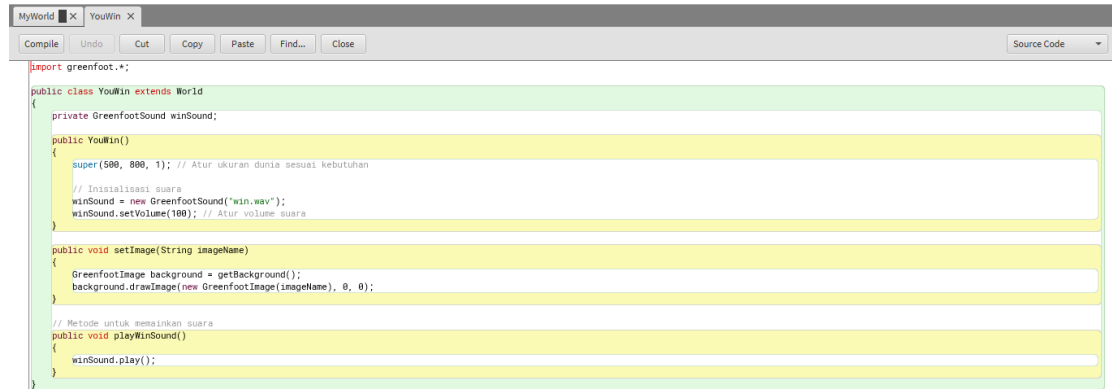
Script di atas adalah bagian dari kode program Java yang menggunakan Greenfoot untuk membuat game sederhana. Berikut ialah penjelasan fungsi-fungsi utamanya.

1. Constructor MyWorld() digunakan untuk membuat dunia permainan dengan ukuran 500x800 sel dengan ukuran 1x1 pixel. Bagian prepare, objek-objek seperti mobil, peluru dan counter untuk skor, HP dan waktu ditambahkan ke dunia permainan.
2. Method act() dijalankan terus-menerus selama permainan berlangsung pada kode yang diberikan. Method ini memanggil 2 method lain yaitu countTime() dan checkGameEnd().
3. Method countTime() untuk menghitung waktu dalam permainan. Waktu terus bertambah dan disajikan dalam detik.
4. Method showTime() untuk menampilkan waktu secara terus-menerus di layar permainan.
5. Method checkGameEnd() untuk memeriksa apakah permainan harus berakhir. Permainan akan berakhir jika waktu sudah mencapai 300 detik atau skor sudah mencapai 100. Jika kondisi terpenuhi, akan muncul tampilan kemenangan yaitu

“YouWin” musik latar belakang dihentikan dan permainan berhenti di method “Greenfoot.stop()”.

6. Method starterd() untuk memulai permainan dan memainkan musik latar belakang “suara.playLoop()”.

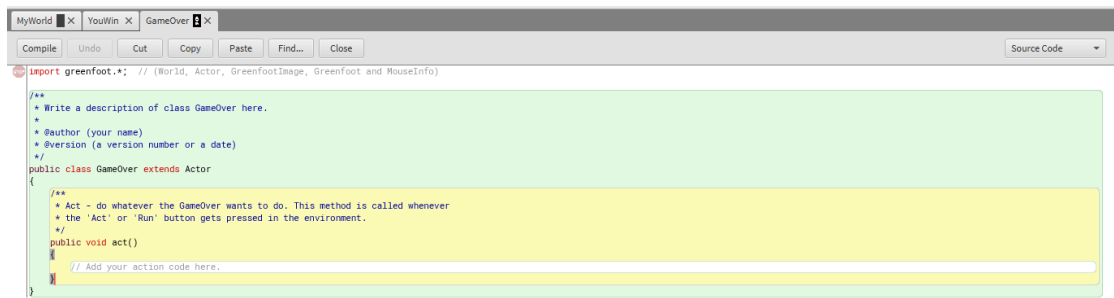
2.2 YouWin



Class “YouWin” merupakan kelas yang mewakili dunia permainan ketika pemain memenangkan permainan. Dan bertanggung jawab untuk menampilkan layar dan memutar suara kemenangan saat berhasil menyelesaikan permainan. Berikut penjelasan fungsi-fungsinya:

1. Constructor YouWin() method ini membuat dunia permainan dengan ukuran 500x800 sel dengan ukuran 1x1 piksel kemudian inisialisasi suara kemenangan menggunakan file suara “win.raw” yang diputar saat pemain memenangkan permainan.
2. Method setImage(String imageName) digunakan untuk mengatur latar belakang dunia permainan dengan gambar yang diberikan dalam bentuk string “imageName”. Metode ini akan menggambar gambar tersebut di seluruh area dunia permainan.
3. Method playWinSound() untuk memainkan suara kemenangan yang sudah diinisialisasi dalam constructor saat pemain memenangkan permainan. Suara akan diputar dengan memakai perintah “winSound.play()”.

2.3 GameOver

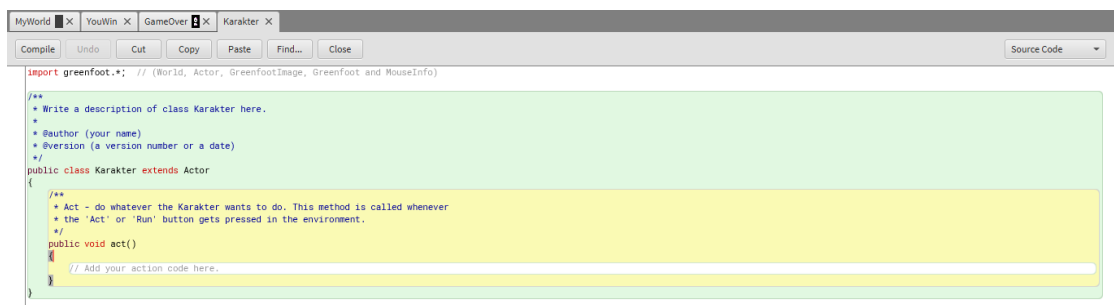


```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class GameOver here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class GameOver extends Actor
{
    /**
     * Act - do whatever the GameOver 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.
    }
}
```

Class GameOver bertanggungjawab untuk menampilkan layar permainan ketika permainan berakhir. Method “act()” tidak memiliki kode aksi apa pun yang dijalankan saat tombol “Act” atau “Run” ditekan di lingkungan permainan. Dalam kondisi ini class tidak melakukan tindakan apapun ketika digunakan dalam permainan. Class ini menampilkan layar permainan telah berakhir atau kalah.

2.4 Karakter

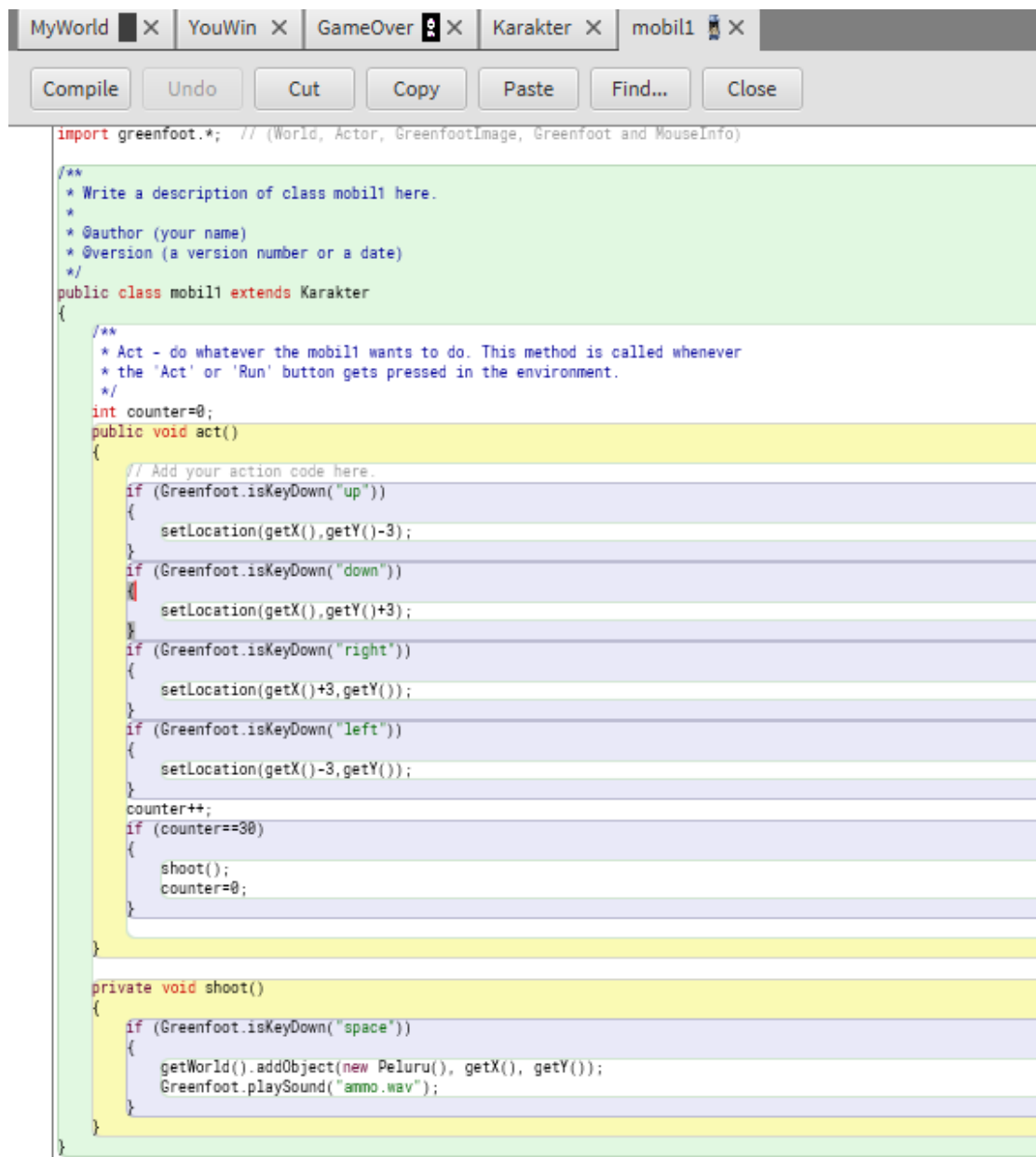


```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * 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.
    }
}
```

Class karakter ialah kelas yang mewakili karakter dalam permainan. Method act() tidak memiliki implementasi aksi apa pun yang dilakukan.

2.5 Mobil1



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

/**
 * Write a description of class mobil1 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class mobil1 extends Karakter
{
    /**
     * Act - do whatever the mobil1 wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    int counter=0;
    public void act()
    {
        // Add your action code here.
        if (Greenfoot.isKeyDown("up"))
        {
            setLocation(getX(),getY()-3);
        }
        if (Greenfoot.isKeyDown("down"))
        {
            setLocation(getX(),getY()+3);
        }
        if (Greenfoot.isKeyDown("right"))
        {
            setLocation(getX()+3,getY());
        }
        if (Greenfoot.isKeyDown("left"))
        {
            setLocation(getX()-3,getY());
        }
        counter++;
        if (counter==30)
        {
            shoot();
            counter=0;
        }
    }

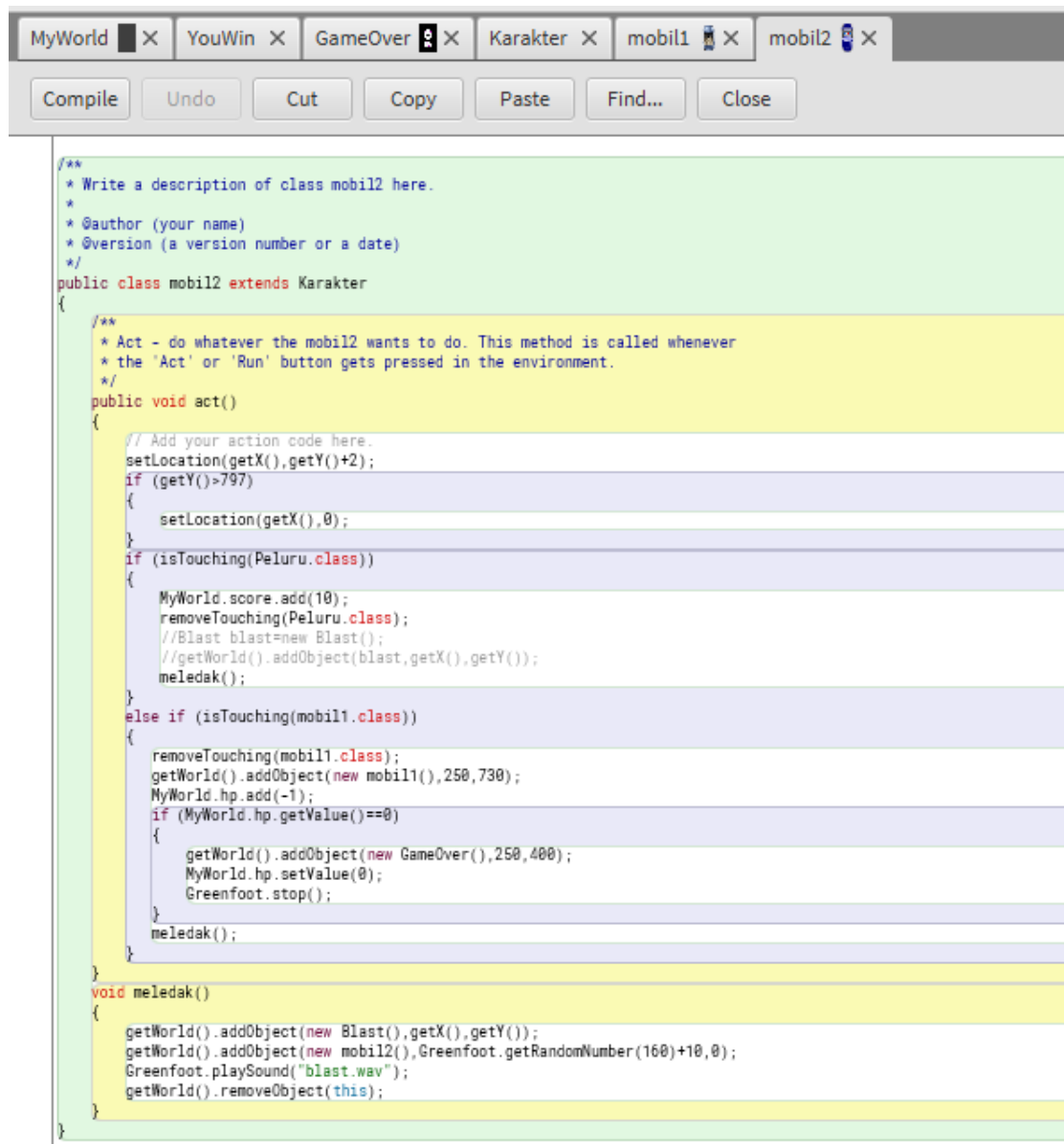
    private void shoot()
    {
        if (Greenfoot.isKeyDown("space"))
        {
            getWorld().addObject(new Peluru(), getX(), getY());
            Greenfoot.playSound("ammo.wav");
        }
    }
}
```

Class mobil1 ialah subkelas dari karakter yang merepresentasikan sebuah mobil dalam permainan. Dalam method “act()” terdapat logika aksi yang dijalankan “Act” atau “Run” ditekan dalam lingkungan permainan. Bertanggung jawab atas pergerakan dan kemampuan menembak objek “peluru” dari mobil tersebut. Terdapat beberapa perintah mengatur perilaku mobil.

1. Pengendalian memakai perintah kondisional “if” untuk menggerakkan mobil ke atas, bawah, kanan atau kiri berdasarkan input yang diberikan. Misal “up” maka akan bergerak ke atas sejauh 3 piksel.

2. Penembakkan yang dimana mobil dapat menembakkan objek “Peluru” saat “space” ditekan. Method “shoot()” dipanggil tiap 30 iterasi “act()” untuk memeriksa apakah tombol “space” ditekan, dan jika ya maka objek “Peluru” akan ditambahkan ke dunia permainan di posisi mobil.

2.6 Mobil2



```
/**
 * Write a description of class mobil2 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class mobil2 extends Karakter
{
    /**
     * Act - do whatever the mobil2 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.
        setLocation(getX(),getY()+2);
        if (getY()>797)
        {
            setLocation(getX(),0);
        }
        if (isTouching(Peluru.class))
        {
            MyWorld.score.add(10);
            removeTouching(Peluru.class);
            //Blast blast=new Blast();
            //getWorld().addObject(blast,getX(),getY());
            meledak();
        }
        else if (isTouching(mobil1.class))
        {
            removeTouching(mobil1.class);
            getWorld().addObject(new mobil1(),250,730);
            MyWorld.hp.add(-1);
            if (MyWorld.hp.getValue()==0)
            {
                getWorld().addObject(new GameOver(),250,400);
                MyWorld.hp.setValue(0);
                Greenfoot.stop();
            }
        }
        meledak();
    }

    void meledak()
    {
        getWorld().addObject(new Blast(),getX(),getY());
        getWorld().addObject(new mobil2(),Greenfoot.getRandomNumber(160)+10,0);
        Greenfoot.playSound("blast.wav");
        getWorld().removeObject(this);
    }
}
```

Class “mobil2” merupakan subkelas dari “Karakter” yang mempresentasikan sebuah mobil lain dalam permainan. Berikut penjelasan fungsinya.

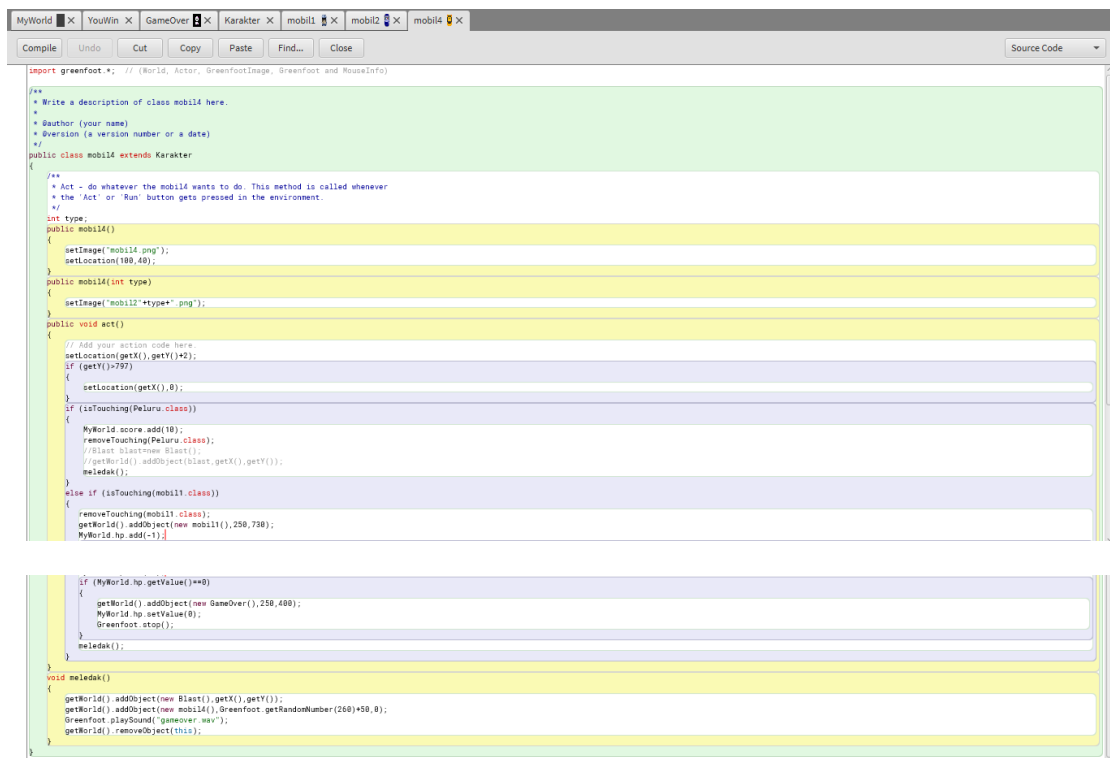
1. Method act() dijalankan secara terus-menerus dalam lingkungan permainan. Kode yang diberikan, mobil ini bergerak ke bawah dengan kecepatan 2 piksel

setiap iterasi. Terdapat kondisi yaitu jika mobil mencapai batas bawah layar di atas posisi 797 maka posisi mobil akan dipindahkan ke atas layar (posisi 0). Jika mobil menyentuh objek “peluru”, skor akan ditambah 10 poin, objek “Peluru” dihapus dan method “meledak()” dipanggil untuk menyisipkan efek ledakan dan jika mobil menyentuh “mobil1”, “mobil1” dihapus dan sebuah “mobil1” baru ditambahkan di tengah atas layar dan nilai HP dikurangi 1. Jika HP mencapai 0 muncul tampilan “GameOver” ditambahkan dan permainan berhenti.

2. Method meledak() bertanggung jawab menyisipkan efek ledakan saat mobil menyentuh “Peluru” atau “mobil1”. Method ini menambahkan objek “Blast” ke dunia permainan, menambahkan mobil baru (“mobil2”) dari atas layar dengan posisi acak di bagian atas layar, memainkan suara ledakan dan menghapus mobil “mobil2” yang saat ini sedang dijalankan.

Berarti logi yang diberikan dalam act(), kelas mobil2 mengatur perilaku mobil kedua dalam permainan, termasuk interaksi dengan objek lain dan penanganan kondisi saat mobil menyentuh objek tertentu.

2.7 Mobil4



```

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

/**
 * Write a description of class mobil4 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class mobil4 extends Karakter
{
    /**
     * Act - do whatever the mobil4 wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    int type;
    public mobil4()
    {
        setImage("mobil4.png");
        setLocation(180,48);
    }
    public mobil4(int type)
    {
        setImage("mobil2"+type+".png");
    }
    public void act()
    {
        // Add your action code here
        setLocation(getX(),getY()+2);
        if (getY()>797)
        {
            setLocation(getX(),0);
        }
        if (isTouching(Peluru.class))
        {
            MyWorld.score.add(10);
            removeTouching(Peluru.class);
            //Blast blast=new Blast();
            //getWorld().addObject(blast,getX(),getY());
            meledak();
        }
        else if (isTouching(mobil1.class))
        {
            removeTouching(mobil1.class);
            getWorld().addObject(new mobil1(),250,738);
            MyWorld.hp.add(-1);
        }
    }

    if (MyWorld.hp.getValue()==0)
    {
        getWorld().addObject(new GameOver(),250,480);
        MyWorld.hp.setValue(0);
        Greenfoot.stop();
    }
    meledak();
}

void meledak()
{
    getWorld().addObject(new Blast(),getX(),getY());
    getWorld().addObject(new mobil4(),Greenfoot.getRandomNumber(260)+50,8);
    Greenfoot.playSound("gameover.wav");
    getWorld().removeObject(this);
}

```

Class “mobil4” adalah subkelas dari “Karakter” yang mewakili jenis mobil lain dalam permainan. Berikut penjelasan fungsinya:

1. Consturctor mobil4() mengatur gambar dan lokasi awal mobil di kordinat 100, 40 pada layar.
2. Constructor mobil4(int type) menerima paramter “type” dan mengatur gambar mobil sesuai dengan tipe yang diberikan.
3. Method act() dijalankan terus-menerus dalam lingkungan permainan mobil ini bergerak ke bawah dengan kecepatan 2 piksel setiap iterasi dan jika mencapai batas bawah layar di atas posisi 797 posisi mobil dipindahkan kembali ke atas layar.
4. Interaksi terdapat kondisi yang memeriksa apakah mobil menyentuh objek “Pelur” atau “mobil1”. Jika mobil menyentuh “Peluru” skor akan ditambah 10 poin, objek “Peluru” dihapus, dan method “meledak()” dipanggil untuk menyisipkan efek ledakan. Jika mobil menyentuh “mobil1”, mobil “mobil1” dihapus dari layar, sebuah “mobil1” baru ditambahkan di tengah atas layar, nilai HP dikurangi satu dan jika HP menacapai 0, tampilan “GameOver” ditambahkan dan permaian dihentikan.
5. Method meledak() menyisipkan efek ledakan saat mobi menyentuh objek “Peluru” atau “mobil1”. Method ini menambahkan objek “Blast” ke dunia permainan, menambahkan mobil baru (“mobil4”) dari atas layar dengan posisi acak, memainkan suara ledakan dan menghapus “mobil4” yang sedang dijalankan.

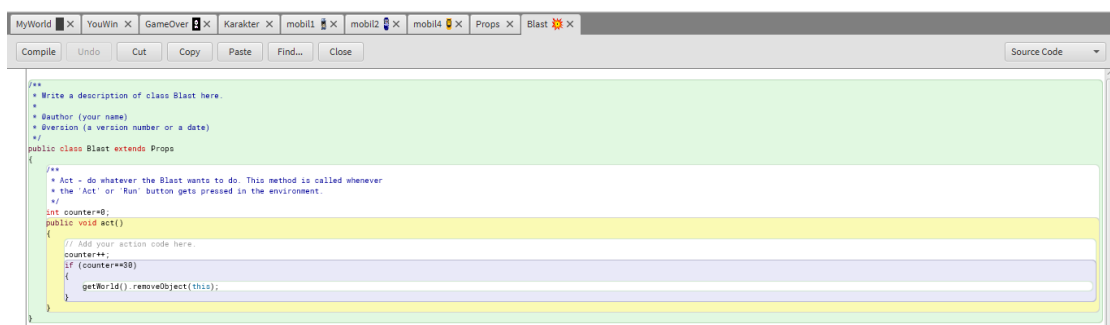
Dengan logika dalam “act()”, kelas “mobil4” mengatur perilaku mobil keempat dalam permainan, termasuk interaksi dengan objel lain dan penanganan kondisi saat mobil menyentuh objek tertentu.

2.8 Props



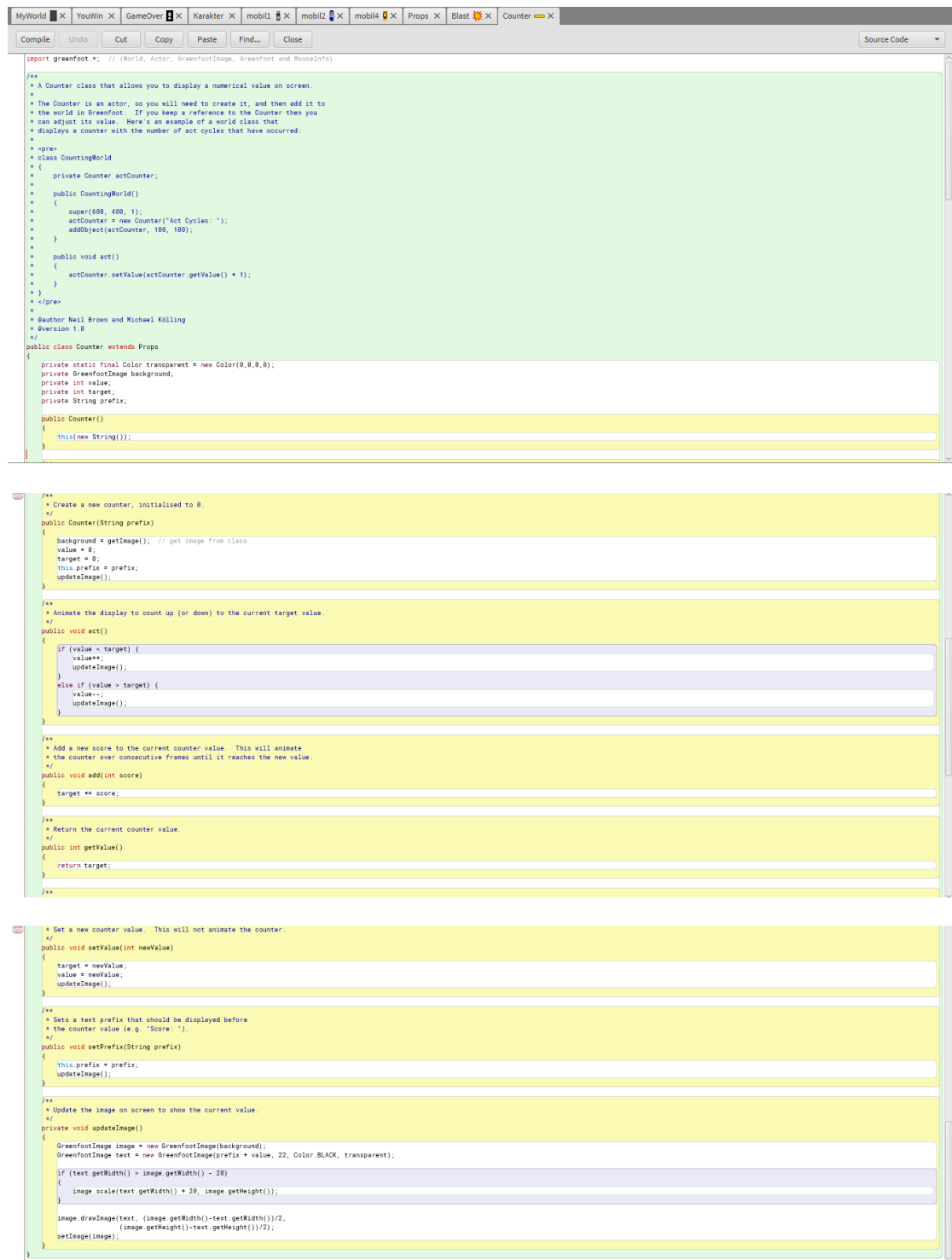
Class “Props” ialah kelas dalam permainan yang mewakili elemen properti atau obyek tertentu yang mungkin digunakan sebagai dekorasi, kendala atau elemen tambahan dalam dunia permainan. Method “act()” untuk mengatur perilaku interaksi dari objek “Props” dalam permainan. Class “Props” memiliki kemampuan menambahkan elemen properti atau objek tambahan ke dalam dunia permainan sesuai kebutuhan yang diinginkan.

2.9 Blast



Class “Blast” merupakan subkelas dari “Props” yang merepresentasikan efek ledakan dalam permainan. Fungsinya untuk menambahkan efek ledakan pada layar dan menghapusnya setelah beberapa iterasi permainan. Method “act()” dijalankan terus-menerus dalam lingkungan permainan. Method variabel “counter” untuk menghitung jumlah iterasi. Setiap kali “act()” dijalankan, variabel “counter” akan bertambah satu. Jika “counter” mencapai nilai 30, objek “Blast” akan dihapus dari layar dengan menggunakan “getWorld().removeObject(this);” Class ini mengontrol durasi efek ledakan sebelum menghapusnya dari world dan memberikan visualisasi singkat dari ledakan dalam permainan.

2.10 Counter



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

/**
 * A Counter class that allows you to display a numerical value on screen.
 *
 * The Counter is an actor, so you will need to create it, and then add it to
 * the world in Greenfoot. If you keep a reference to the Counter then you
 * can adjust its value. Here's an example of a world class that
 * displays a counter with the number of act cycles that have occurred:
 *
 * <pre>
 * class CountingWorld
 * {
 *     private Counter actCounter;
 *
 *     public CountingWorld()
 *     {
 *         super(600, 400, 1);
 *         actCounter = new Counter("Act Cycles: ");
 *         addObject(actCounter, 100, 100);
 *     }
 *
 *     public void act()
 *     {
 *         actCounter.setValue(actCounter.getValue() + 1);
 *     }
 * }
 * </pre>
 *
 * Author: Neil Brown and Michael Kölling
 * Version: 1.0
 */
public class Counter extends Props
{
    private static final Color transparent = new Color(0,0,0,0);
    private GreenfootImage background;
    private int value;
    private int target;
    private String prefix;

    public Counter()
    {
        this(new String());
    }

    /**
     * Create a new counter, initialised to 0.
     */
    public Counter(String prefix)
    {
        background = getImage(); // get image from class
        value = 0;
        target = 0;
        this.prefix = prefix;
        updateImage();
    }

    /**
     * Animate the display to count up (or down) to the current target value.
     */
    public void act()
    {
        if (value < target) {
            value++;
            updateImage();
        }
        else if (value > target) {
            value--;
            updateImage();
        }
    }

    /**
     * Add a new score to the current counter value. This will animate
     * the counter over consecutive frames until it reaches the new value.
     */
    public void add(int score)
    {
        target += score;
    }

    /**
     * Return the current counter value.
     */
    public int getValue()
    {
        return target;
    }

    /**
     * Set a new counter value. This will not animate the counter.
     */
    public void setValue(int newValue)
    {
        target = newValue;
        value = newValue;
        updateImage();
    }

    /**
     * Sets a text prefix that should be displayed before
     * the counter value (e.g. "Score: ").
     */
    public void setPrefix(String prefix)
    {
        this.prefix = prefix;
        updateImage();
    }

    /**
     * Update the image on screen to show the current value.
     */
    private void updateImage()
    {
        GreenfootImage image = new GreenfootImage(background);
        GreenfootImage text = new GreenfootImage(prefix + value, 22, Color.BLACK, transparent);

        if (text.getWidth() > image.getWidth() - 20)
        {
            image.scale(text.getWidth() * 20, image.getHeight());
        }

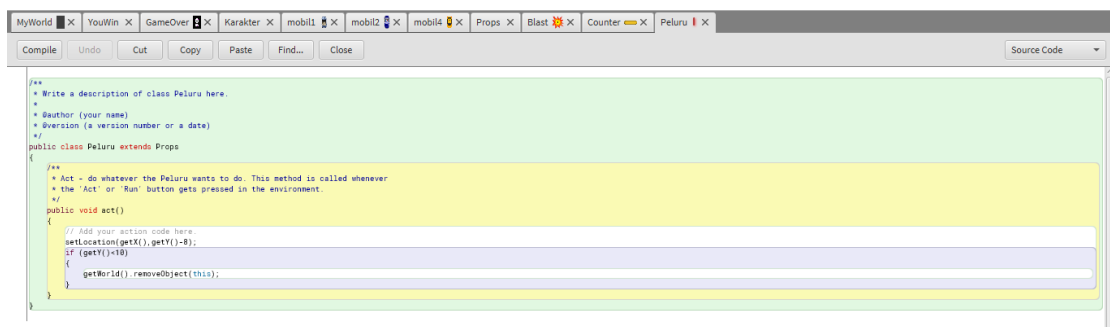
        image.drawImage(text, (image.getWidth() - text.getWidth()) / 2,
            (image.getHeight() - text.getHeight()) / 2);
        setImage(image);
    }
}
```

Class “Counter” merupakan subkelas dari “Props” berfungsi penghitung atau petunjuk nilai numerik yang ditampilkan. Berikut penjelasan fungsi-fungsinya:

1. Counter() konstruktor default yang membuat objek “Counter” dengan nilai awal 0.

2. Counter(String prefix) untuk membuat objek “Counter” dengan nilai awal 0 dan menambahkan teks awalan sesuai yang diberikan.
3. Act() method dijalankan secara terus-menerus. Fungsinya untuk menyesuaikan nilai yang ditampilkan agar mencapai nilai target yang diinginkan secara animasi (perlahan bertambah atau berkurang).
4. Add(int score) method ini menambahkan nilai tertentu ke dalam nilai counter saat ini. Nilai akan diubah secara animasi dalam beberapa iterasi permainan hingga mencapai nilai baru.
5. GetValue() untuk mengembalikan nilai yang ditampilkan saat ini pada counter.
6. SetValue(int newValue) untuk mengatur nilai counter ke nilai baru yang diberikan tanpa melakukan animasi.
7. setPrefix(String prefix) untuk mengatur teks awalan yang akan ditampilkan sebelum nilai counter.
8. UpdateImage() bertanggung jawab untuk memperbarui tampilan gambar yang menampilkan nilai counter beserta teks awalan yang diinginkan pada layar.

2.11 Peluru



Class “Peluru” merupakan subkelas dari “Props” yang mempresentasikan perilaku peluru. Method act() ini berisi aksi yang dilakukan oleh objek “Peluru” saat permainan berjalan. Secara khusus method ini mengatur pergerakan peluru dengan mengubah posisi Y (koordinat vertikal) dari objek. Peluru akan bergerak ke atas dengan kecepatan 8 Piksel setiap aksi “act”. Jika peluru mencapai batas layar (posisi Y kurang dari 10 piksel) maka objek peluru akan dihapus.

BAB III

PENUTUP

3.1 Kesimpulan

Proyek ini membuktikan pentingnya penerapan prinsip-prinsip PBO dalam pengembangan game, mengasah keterampilan pemrograman dan menghadirkan pengalaman praktis dalam memahami elemen-elemen kunci pengembangan permainan.

3.2 Saran

Dalam pengembangan game ini penting untuk mengeksplorasi beberapa cara-cara lain yang lebih mangkus dan terbaik. Memanfaatkan konsep pewarisan dan elemen-elemen lainnya yang telah disediakan Greenfoot secara optimal yang bisa menghasilkan game yang lebih menarik, interaktif dan mudah dimengerti pengguna.