

TUTORIAL MEMBUAT GAME KUMBANG ADVENTURE MENGUNAKAN GREENFOOT



Disusun oleh:

Dedi Bintang Pamungkas

18030020 (TF A)

PRODI TEKNIK INFORMATIKA

STT ADISUTJIPTO YOGYAKARTA 2018

A. Pengertian



Greenfoot merupakan lingkungan pengembangan Java (Java development environment) yang bersifat interaktif dan didesain untuk tujuan pendidikan pada level SMA dan S1. Dengan program ini, seseorang dapat mengembangkan aplikasi grafis dua dimensi, seperti simulasi dan game interaktif.

Greenfoot dikembangkan oleh University of Kent dan La Trobe University, dengan dukungan dari Oracle. Software ini gratis, dirilis di bawah lisensi GPL. Greenfoot tersedia pada banyak platform, seperti MS Windows, Mac OS X, Linux, Sun Solaris, dan beberapa JVM terbaru.

Dengan greenfoot, seorang siswa/mahasiswa dapat belajar pemrograman berorientasi objek menggunakan Java. Mereka dapat menciptakan 'actor' dalam suatu 'world' untuk membangun game, simulasi, dan program-program grafis lainnya.

Greenfoot bersifat visual dan interaktif. Perangkat visualisasi dan interaksi dibangun di dalam environment tersebut. Actor diprogram menggunakan kode Java yang standar, sehingga memberikan pengalaman pemrograman berbasis teks yang bersifat tradisional dengan eksekusi visual.

Antarmuka yang digunakan sepenuhnya menggunakan konsep IDE, dimana didalamnya terdapat manajemen proyek, auto-completion, syntax highlighting, dan tool-tool lain yang biasanya terdapat pada kebanyakan IDE. Hal ini memungkinkan seorang pemrogram mempublikasikan pekerjaannya baik secara online maupun offline. Meskipun demikian, antarmukanya didesain untuk tetap sederhana dan mudah digunakan, sehingga cocok untuk para pemula.

Greenfoot digunakan oleh ribuan institusi di seluruh dunia. Program ini memberikan peralihan yang mudah ke environment lainnya, semacam BlueJ dan IDE-IDE profesional yang lain.

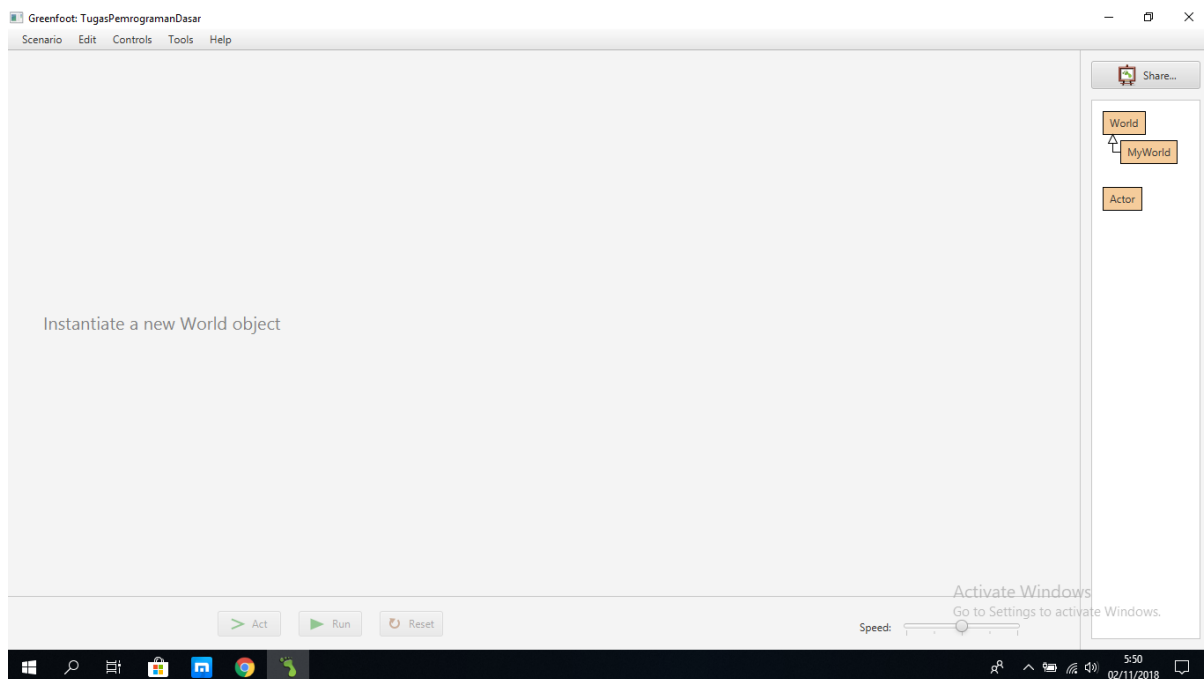
Program dalam greenfoot ditulis dengan bahasa Java standar, yang merupakan salah satu bahasa utama dalam dunia akademis dan industri. Software ini didesain agar cukup mudah bagi pemula. Dengan digunakannya Java, memungkinkan seorang programmer membuat aplikasi yang impresif, fleksibel, dan canggih.

Software ini memang cocok bagi siswa/mahasiswa yang ingin belajar dan memperdalam baik secara konsep maupun praktik pemrograman berorientasi obyek. Software tersebut dapat didownload pada alamat ini <http://www.greenfoot.org/download> yang tersedia dalam berbagai platform sistem operasi.

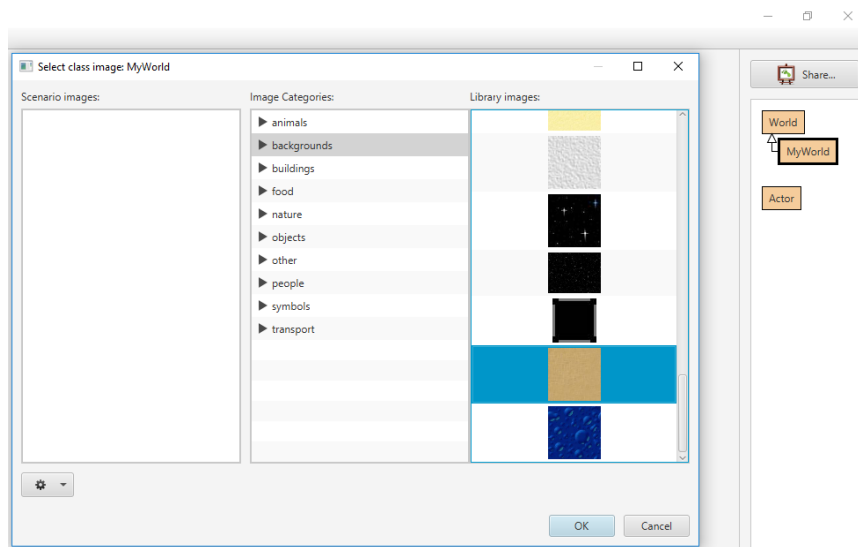
B. Langkah kerja

1. Background

Ini adalah tampilan awal dari Greenfoot, langkah pertama kita akan membuat background dari game yang akan di buat



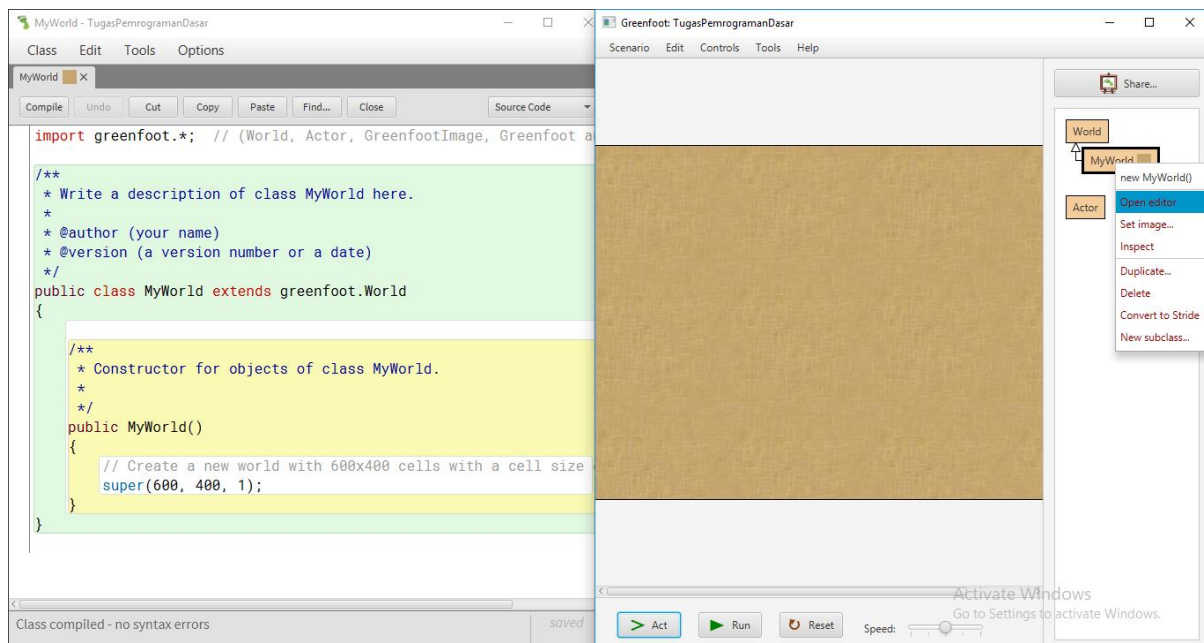
Caranya klik kanan pada class “MyWorld” => pilih setImage => pilih Background yang anda inginkan kemudian klik “OK”



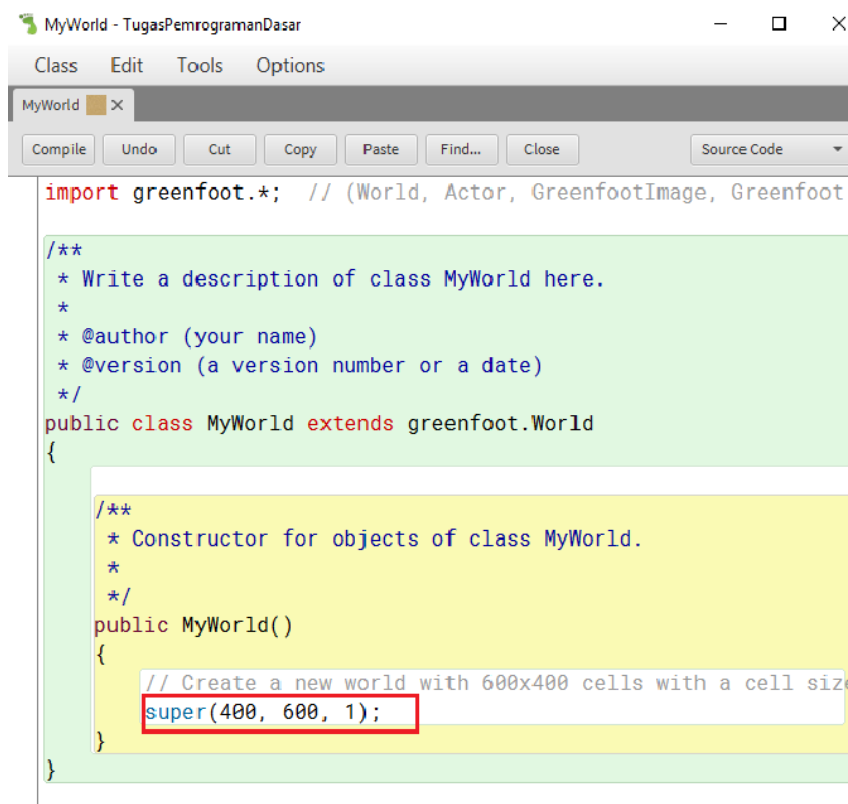
Klik kanan pada class “MyWorld” dan pilih new MyWorld agar background akan terganti sesuai yang kita inginkan



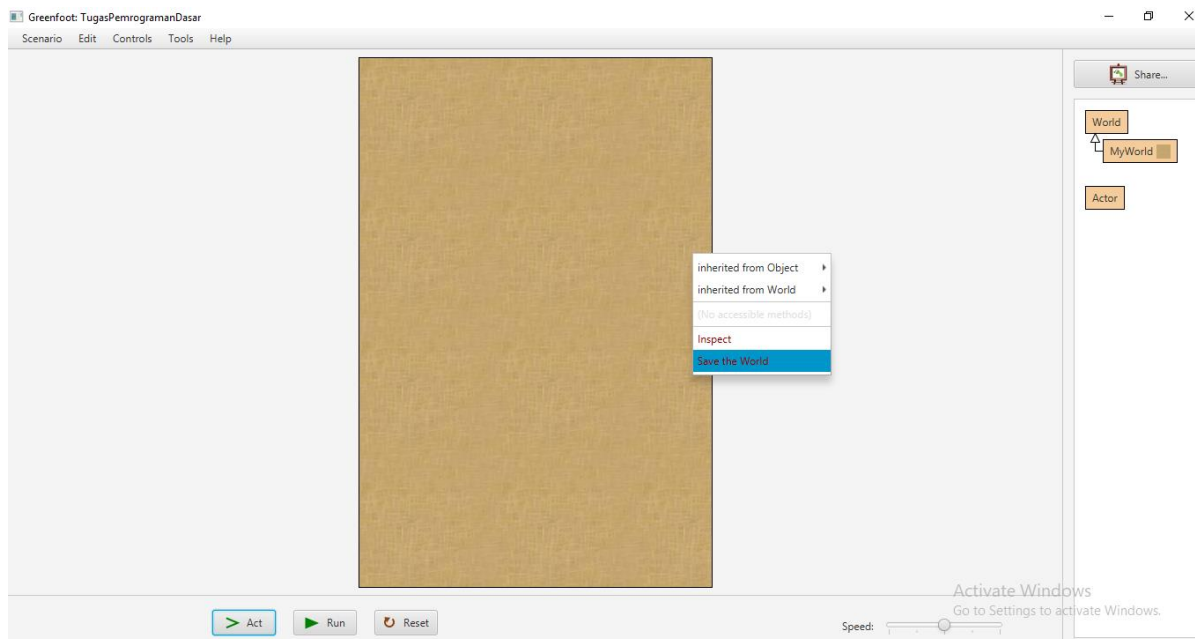
Disini saya ingin mengubah ukuran latar sesuai yang saya rencanakan, yang awal default nya sebesar 600x400, dengan cara klik kanan class “MyWorld” pilih “Open Editor”



Dan ubah seperti yang bergaris merah dibawah, disini saya akan ubah menjadi 400x600 klik => Compile

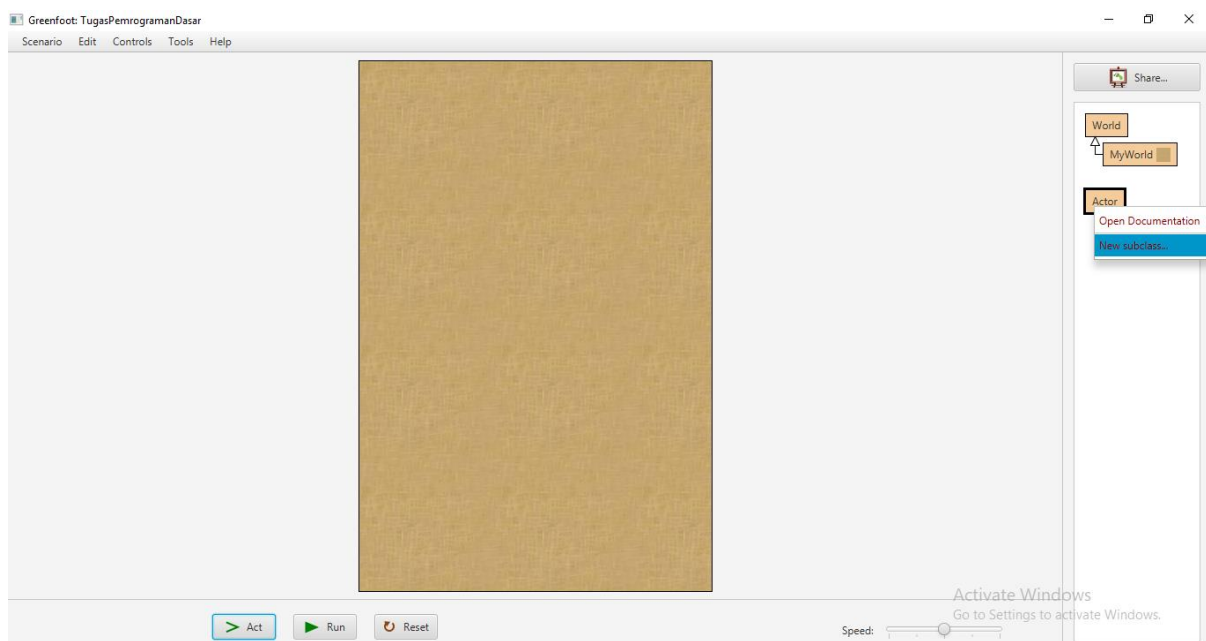


Maka tampilan nya akan seperti di bawah ini, jangan lupa untuk di “Save” dengan cara klik kanan pada latar background dan pilih “Save the World”

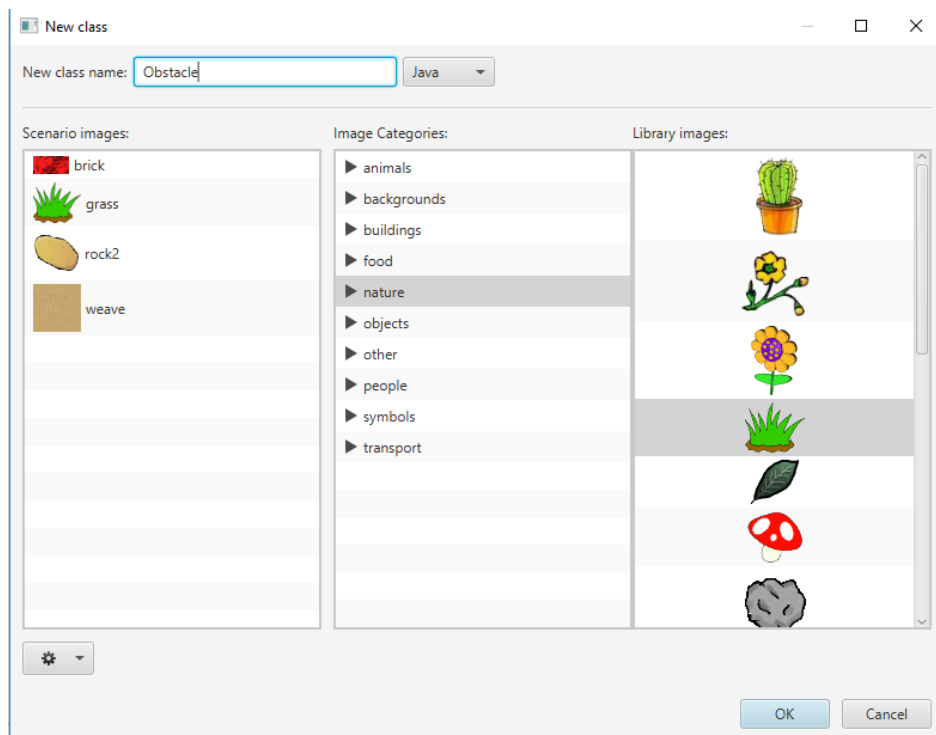


2. Actor

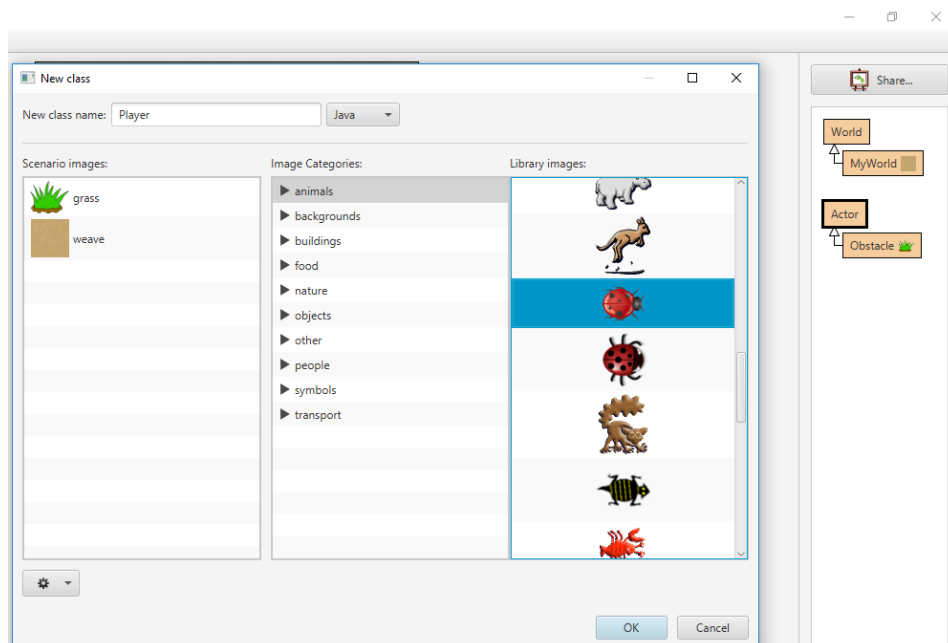
Pada tahap kedua ini saya akan menambahkan beberapa Actor yang nantinya akan berbeperan dalam game yang di buat, dengan cara klik kanan pada class “Actor” => New subclass

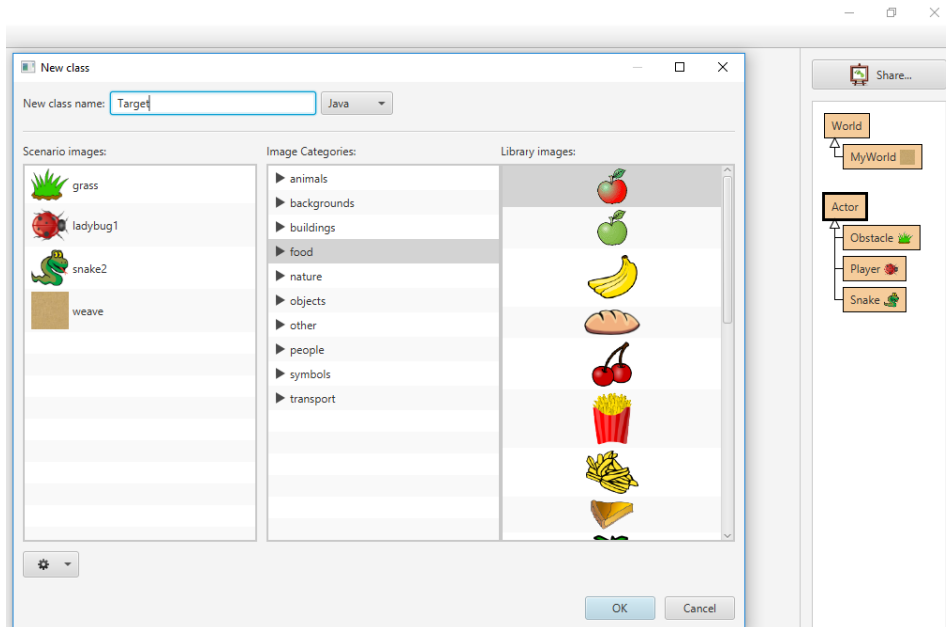
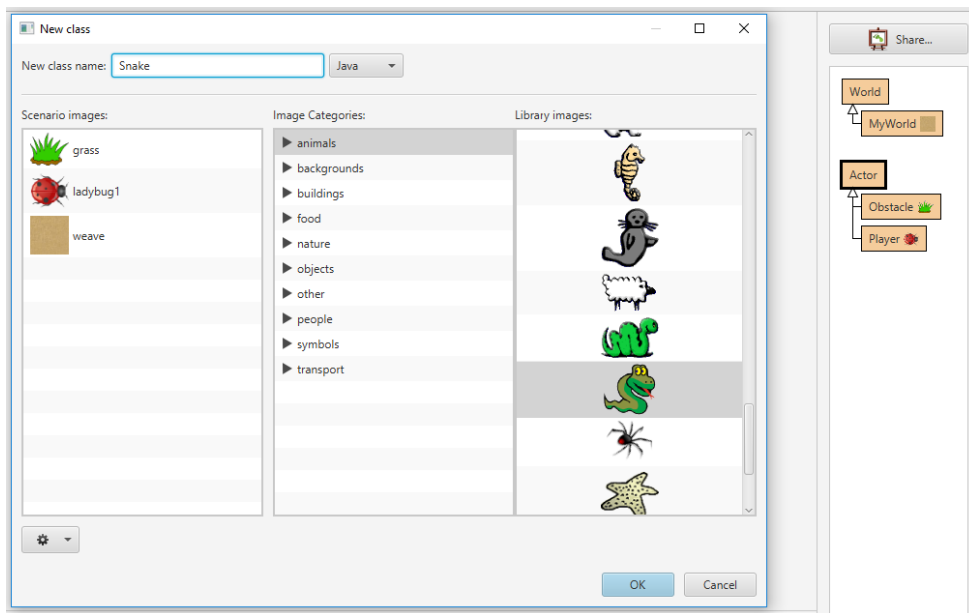


Aktor pertama yang dibuat yaitu "Obstacle" disini saya menggunakan actor Rumput, berilah nama pada actor yang dibuat dan pilih "OK"



Lakukan langkah yang sama untuk Aktor yang lainnya dengan karakter yang sesuai anda inginkan dan jangan lupa beri nama kemudian klik "OK"





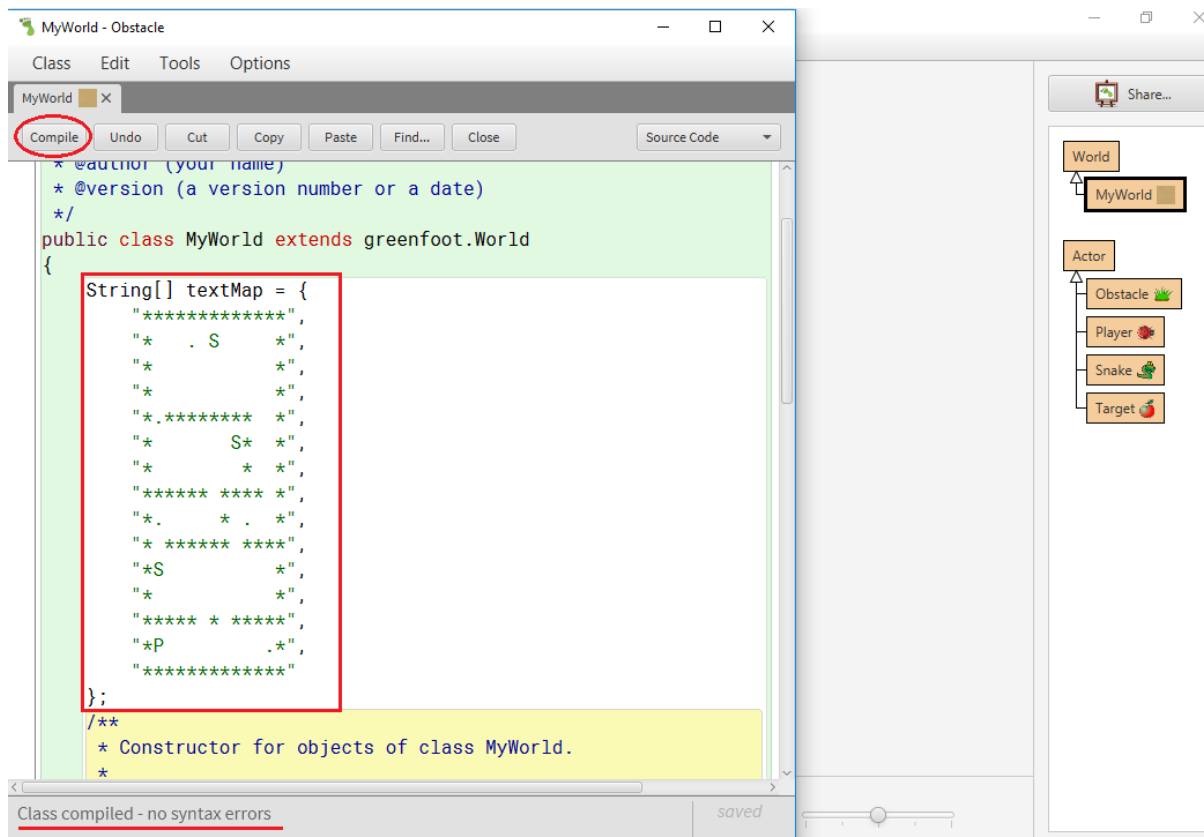
Apabila semua aktor telah dibuat selanjutnya kita akan memposisikan aktor-aktor tersebut dalam sebuah permainan letak sesuai ide permainan kita, caranya Open editor pada class MyWorld => Masukkan perintah seperti gambar di bawah ini,

tanda (*) untuk aktor "Obstacle"

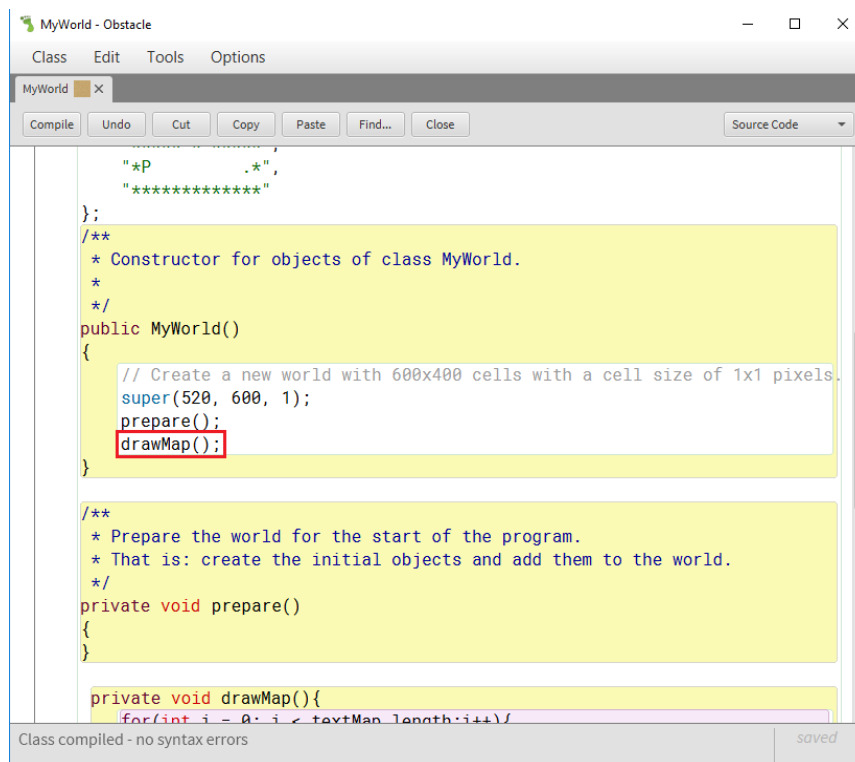
tanda (.) untuk aktor "Target"

tanda (S) untuk "Snake"

tanda (P) untuk "Player"



Buatkan variable “drawMap” yang nantinya berfungsi untuk memanggil aktor dari masing-masing class dan akan memposisikannya dalam permainan yang tadinya telah dibuat



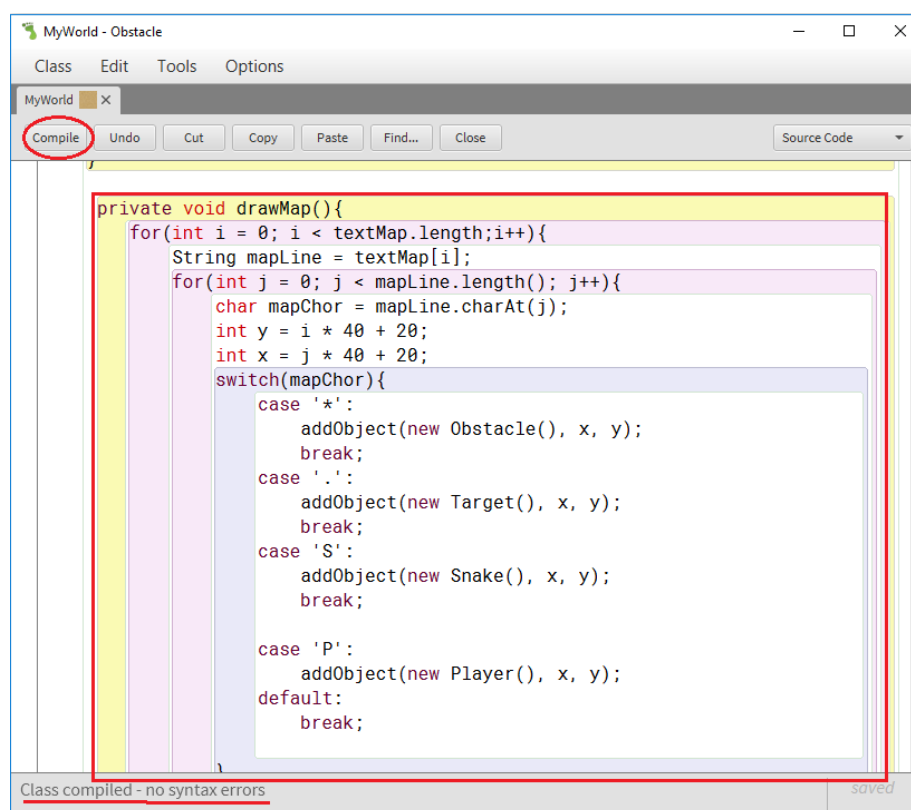
```
MyWorld - Obstacle
Class Edit Tools Options
MyWorld x
Compile Undo Cut Copy Paste Find... Close Source Code
/*
 * Constructor for objects of class MyWorld.
 */
public MyWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
    super(520, 600, 1);
    prepare();
    drawMap();
}

/**
 * Prepare the world for the start of the program.
 * That is: create the initial objects and add them to the world.
 */
private void prepare()
{
}

private void drawMap(){
    for(int i = 0; i < textMap.length; i++){
        String mapLine = textMap[i];
        for(int j = 0; j < mapLine.length(); j++){
            char mapChor = mapLine.charAt(j);
            int y = i * 40 + 20;
            int x = j * 40 + 20;
            switch(mapChor){
                case '*':
                    addObject(new Obstacle(), x, y);
                    break;
                case '.':
                    addObject(new Target(), x, y);
                    break;
                case 'S':
                    addObject(new Snake(), x, y);
                    break;
                case 'P':
                    addObject(new Player(), x, y);
                    break;
                default:
                    break;
            }
        }
    }
}
```

Class compiled - no syntax errors saved

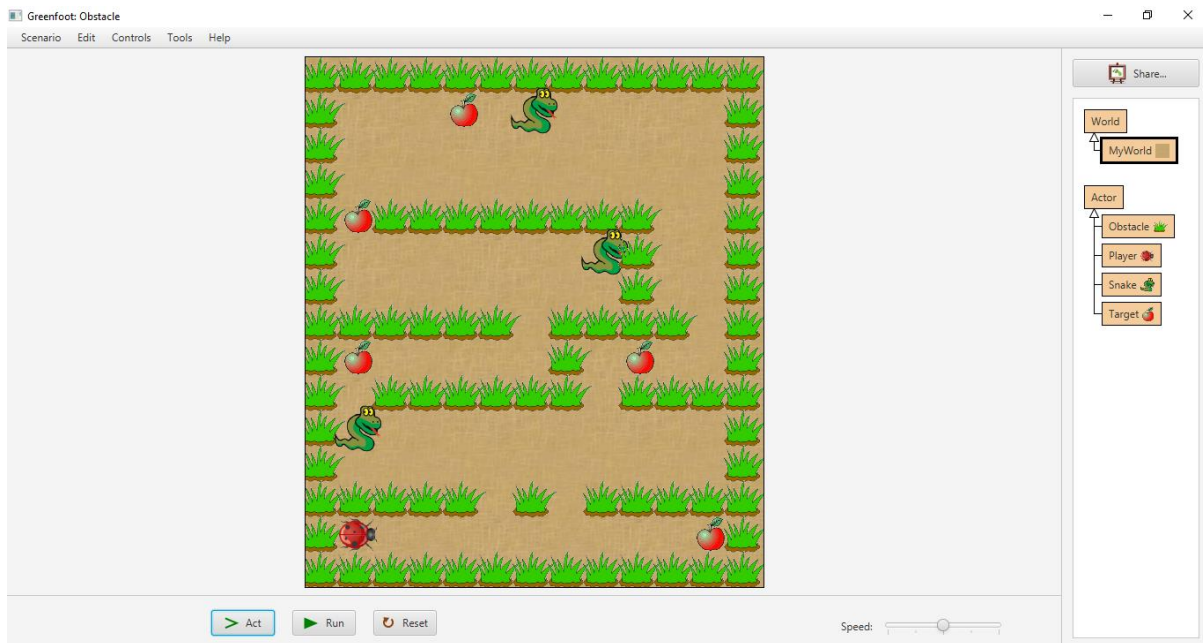
Masukkan skrip seperti gambar dibawah ini, setelah itu “Compile” dan pastikan tidak ada syntax yang error



```
MyWorld - Obstacle
Class Edit Tools Options
MyWorld x
Compile Undo Cut Copy Paste Find... Close Source Code
private void drawMap(){
    for(int i = 0; i < textMap.length; i++){
        String mapLine = textMap[i];
        for(int j = 0; j < mapLine.length(); j++){
            char mapChor = mapLine.charAt(j);
            int y = i * 40 + 20;
            int x = j * 40 + 20;
            switch(mapChor){
                case '*':
                    addObject(new Obstacle(), x, y);
                    break;
                case '.':
                    addObject(new Target(), x, y);
                    break;
                case 'S':
                    addObject(new Snake(), x, y);
                    break;
                case 'P':
                    addObject(new Player(), x, y);
                    break;
                default:
                    break;
            }
        }
    }
}
```

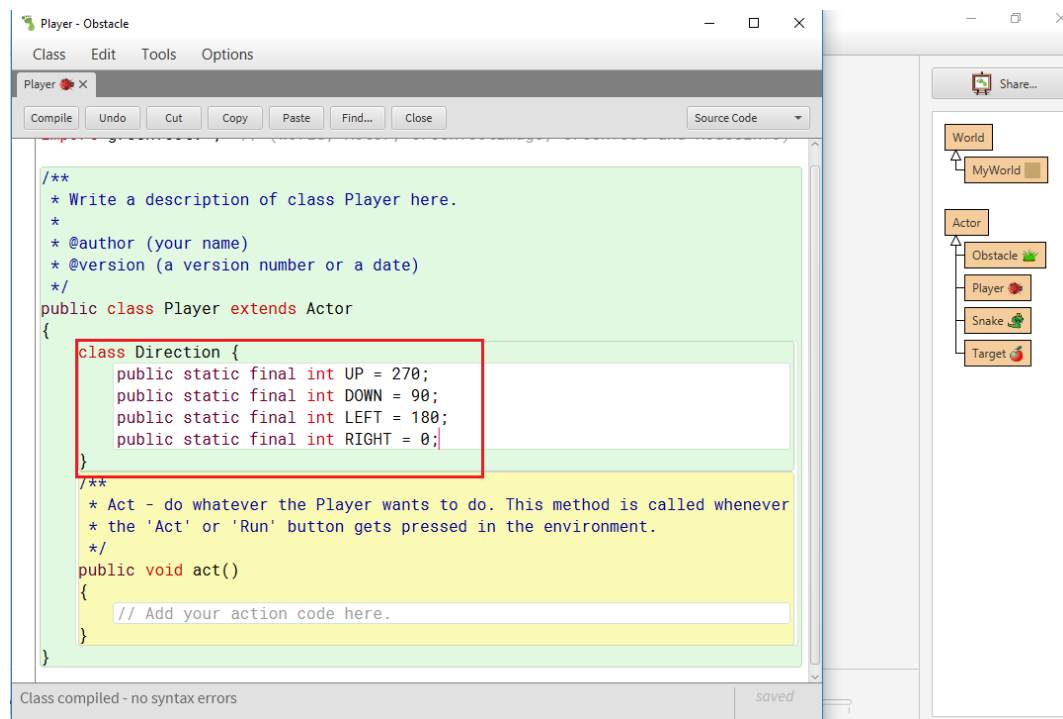
Class compiled - no syntax errors saved

Apabila berhasil maka tampilan nya akan seperti gambar di bawah ini

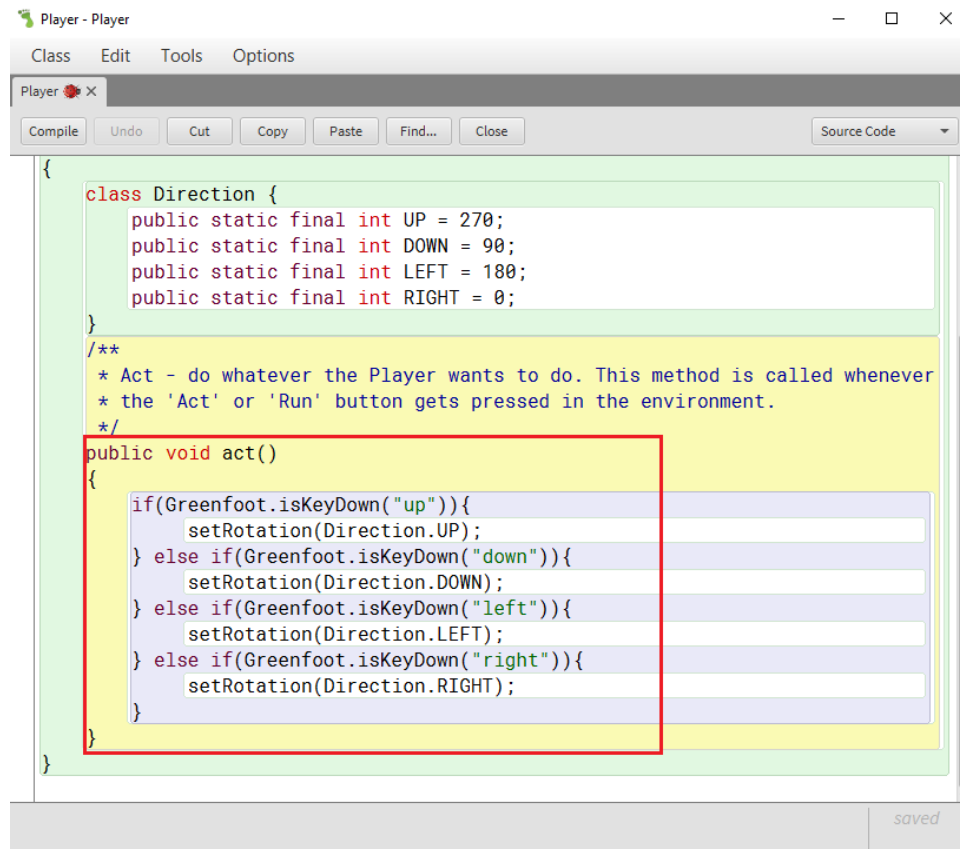


3. Player

Langkah ketiga kita akan memasukkan perintah pada aktor Player agar dapat bergerak sesuai yang kita inginkan, pertama masukkan perintah “class Direction” seperti gambar dibawah ini ini berfungsi agar aktor Player dapat menghadap ke arah (atas, bawah, kanan, kiri)



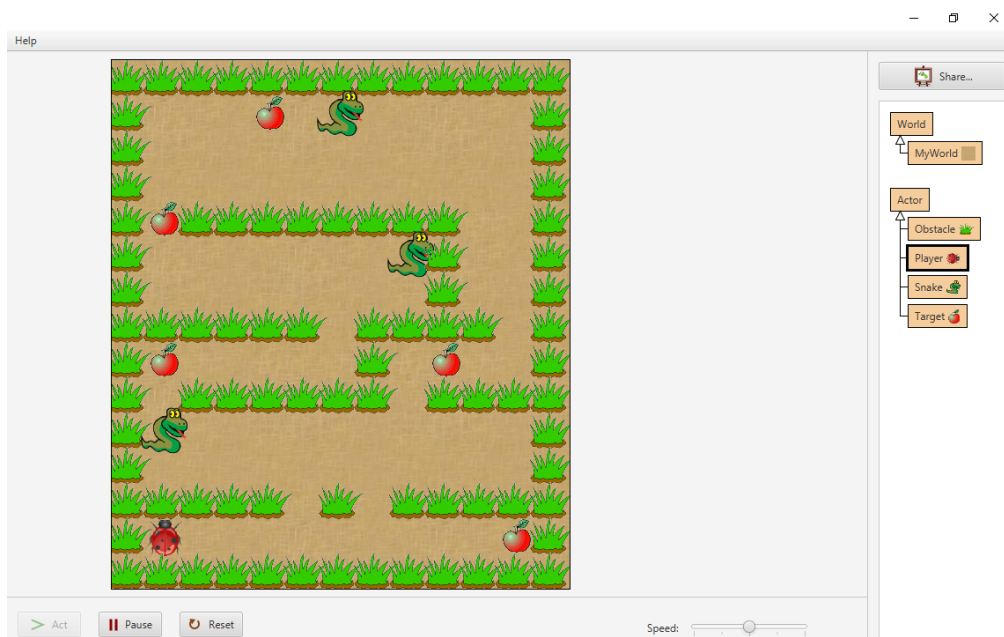
Lanjut perintah untuk menggerakkan aktor menggunakan tombol panah (atas, bawah, kanan, kiri) pada keyboard, perintah nya seperti gambar dibawah ini yang bertanda merah



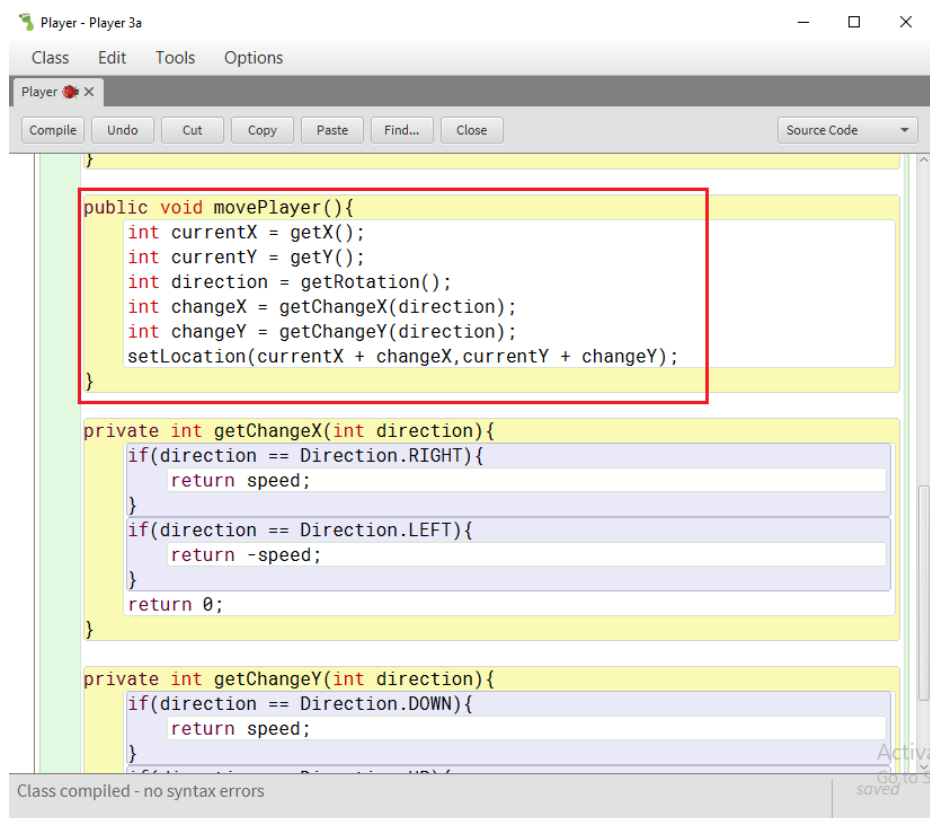
```
{
    class Direction {
        public static final int UP = 270;
        public static final int DOWN = 90;
        public static final int LEFT = 180;
        public static final int RIGHT = 0;
    }

    /**
     * Act - do whatever the Player wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.isKeyDown("up")){
            setRotation(Direction.UP);
        } else if(Greenfoot.isKeyDown("down")){
            setRotation(Direction.DOWN);
        } else if(Greenfoot.isKeyDown("left")){
            setRotation(Direction.LEFT);
        } else if(Greenfoot.isKeyDown("right")){
            setRotation(Direction.RIGHT);
        }
    }
}
```

Jika sudah coba di compile dan lihat hasilnya, apabila aktor bisa menghadap kearah sesuai tombol panah keyboard berarti perintah yang dimasukkan berhasil



Kedua gambar dibawah ini adalah skrip untuk menggerakkan aktor agar dapat berjalan



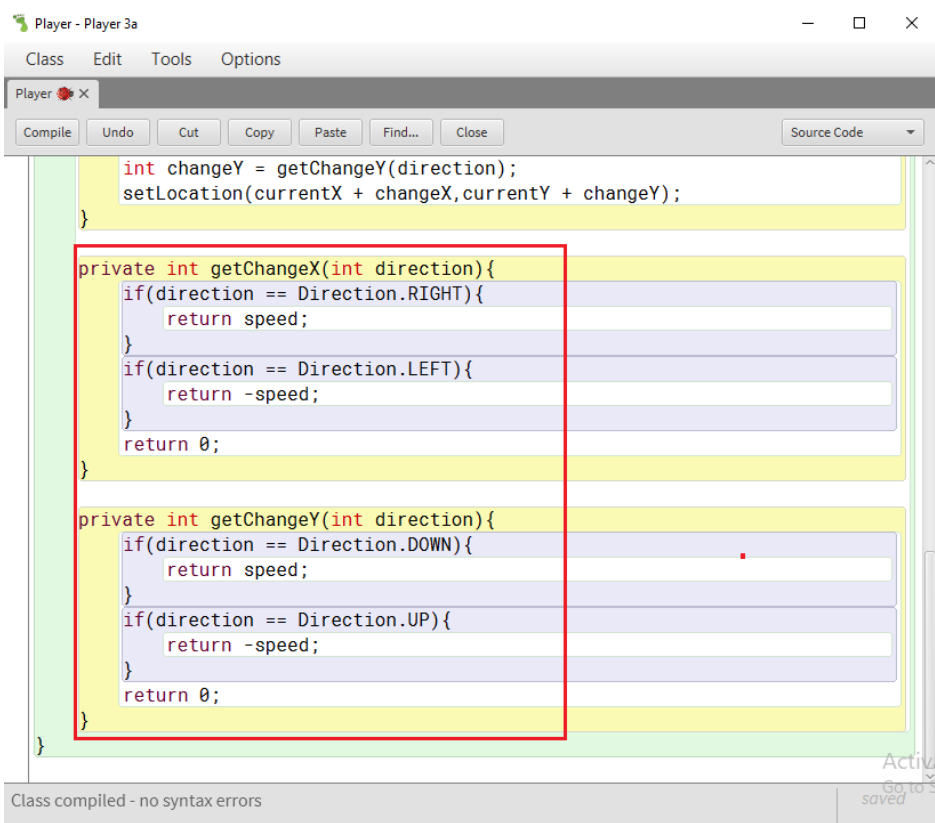
The screenshot shows a Java IDE window titled "Player - Player 3a". The code editor displays the following code:

```
public void movePlayer(){
    int currentX = getX();
    int currentY = getY();
    int direction = getRotation();
    int changeX = getChangeX(direction);
    int changeY = getChangeY(direction);
    setLocation(currentX + changeX, currentY + changeY);
}

private int getChangeX(int direction){
    if(direction == Direction.RIGHT){
        return speed;
    }
    if(direction == Direction.LEFT){
        return -speed;
    }
    return 0;
}

private int getChangeY(int direction){
    if(direction == Direction.DOWN){
        return speed;
    }
    if(direction == Direction.UP){
        return -speed;
    }
    return 0;
}
```

The status bar at the bottom indicates "Class compiled - no syntax errors".



The screenshot shows the same Java IDE window, but with the code scrolled down to show the helper methods:

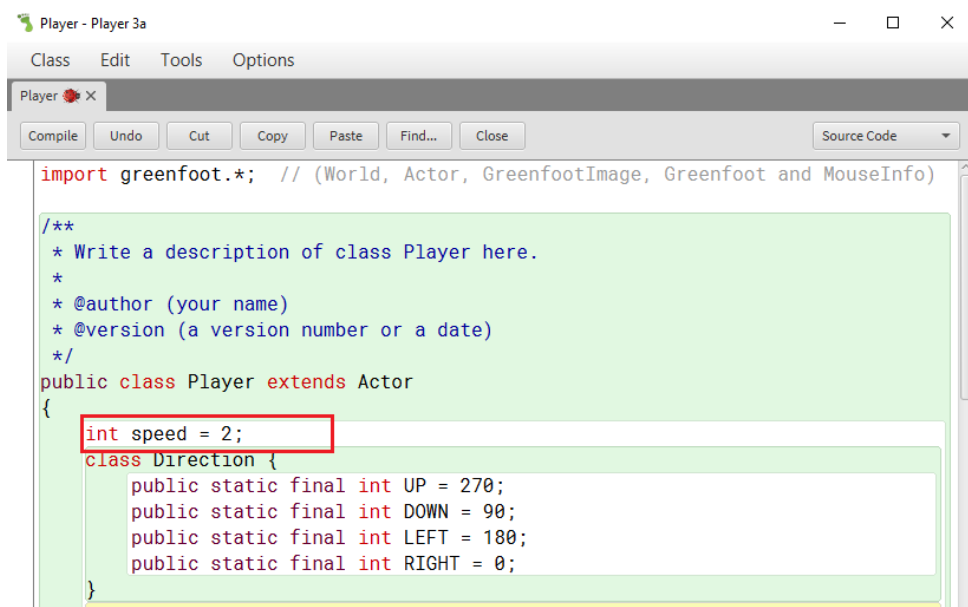
```
int changeY = getChangeY(direction);
setLocation(currentX + changeX, currentY + changeY);
}

private int getChangeX(int direction){
    if(direction == Direction.RIGHT){
        return speed;
    }
    if(direction == Direction.LEFT){
        return -speed;
    }
    return 0;
}

private int getChangeY(int direction){
    if(direction == Direction.DOWN){
        return speed;
    }
    if(direction == Direction.UP){
        return -speed;
    }
    return 0;
}
```

The status bar at the bottom indicates "Class compiled - no syntax errors".

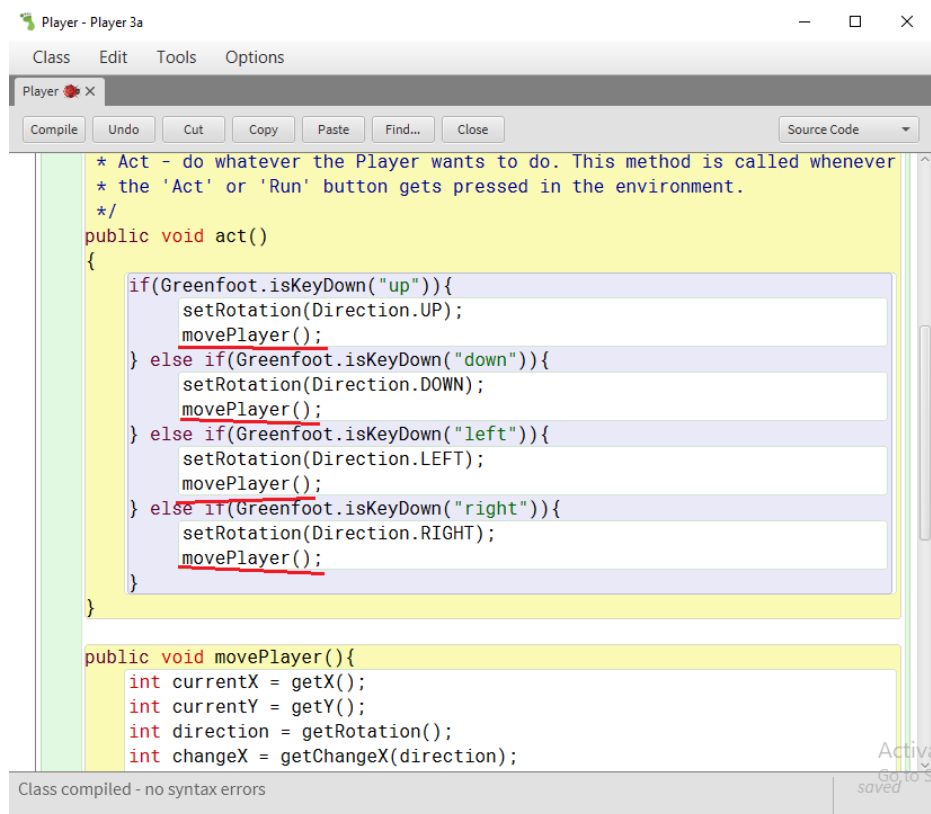
Tambahkan variable speed, kecepatan berjalan nya aktor



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

/**
 * Write a description of class Player here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Player extends Actor
{
    int speed = 2;
    class Direction {
        public static final int UP = 270;
        public static final int DOWN = 90;
        public static final int LEFT = 180;
        public static final int RIGHT = 0;
    }
}
```

Terkahir kombinasikan variable movePlayer ke dalam variable act, agar aktor dapat berjalan sesuai rotasi arah aktor tersebut, setelah itu di “Compile” dan lihat hasilnya



```

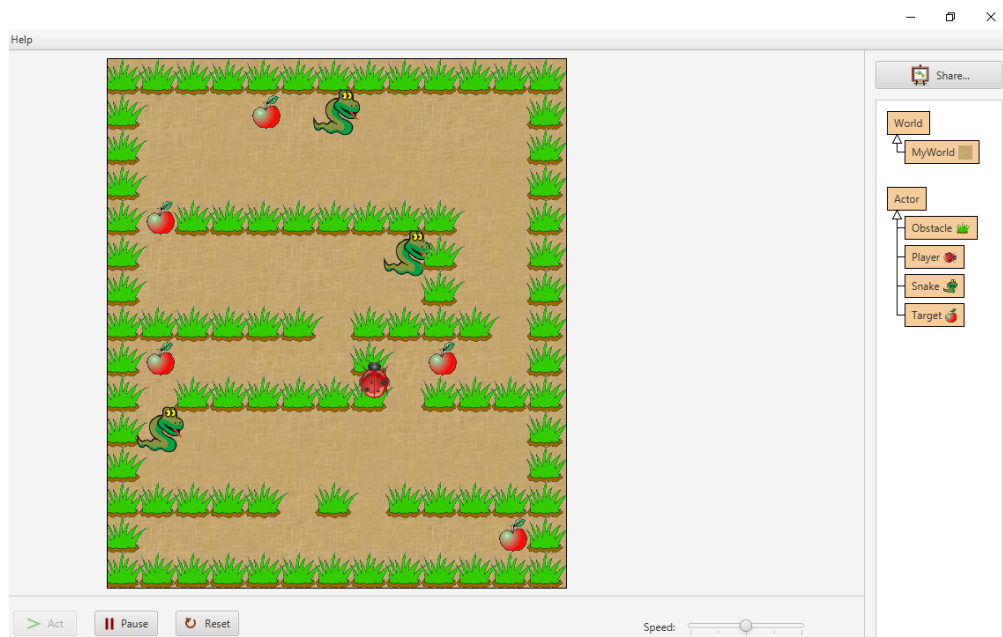
 * Act - do whatever the Player wants to do. This method is called whenever
 * the 'Act' or 'Run' button gets pressed in the environment.
 */
public void act()
{
    if(Greenfoot.isKeyDown("up")){
        setRotation(Direction.UP);
        movePlayer();
    } else if(Greenfoot.isKeyDown("down")){
        setRotation(Direction.DOWN);
        movePlayer();
    } else if(Greenfoot.isKeyDown("left")){
        setRotation(Direction.LEFT);
        movePlayer();
    } else if(Greenfoot.isKeyDown("right")){
        setRotation(Direction.RIGHT);
        movePlayer();
    }
}

public void movePlayer(){
    int currentX = getX();
    int currentY = getY();
    int direction = getRotation();
    int changeX = getChangeX(direction);
}
```

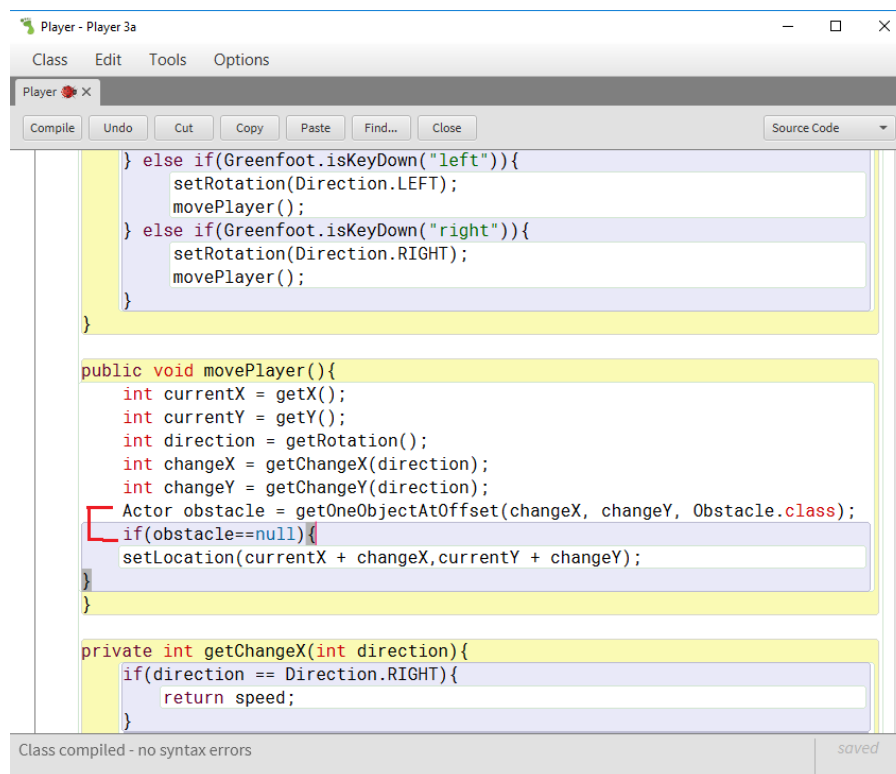
Class compiled - no syntax errors

4. Obstacle

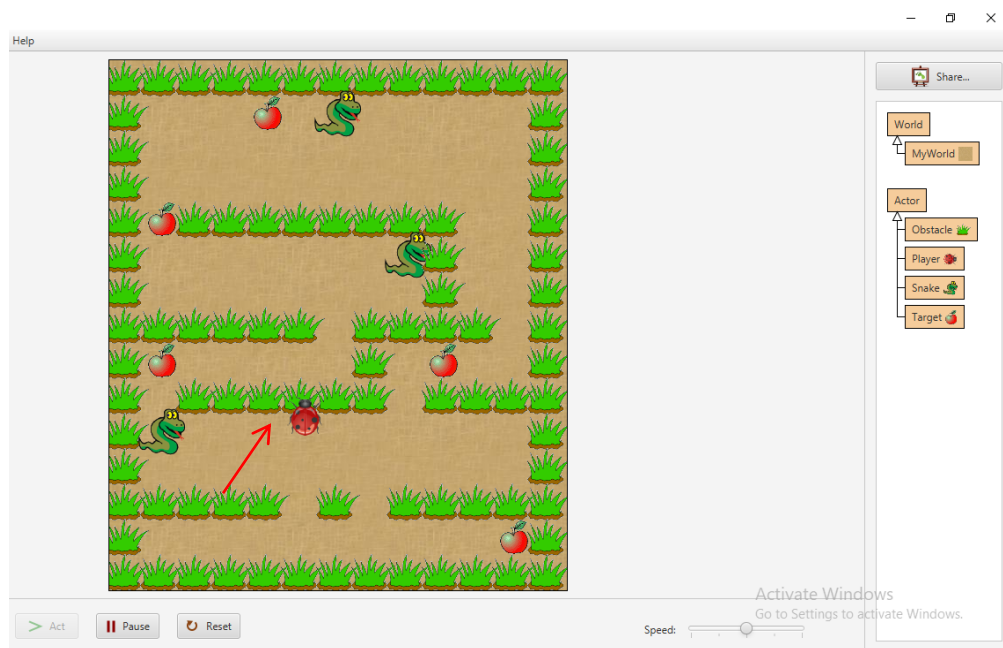
Langkah ini kita akan membuat aktor player agar tidak dapat melewati Obstacle seperti pada gambar dibawah ini, jadi aktor player hanya dapat menyentuh obstacle dari sisi bagian luar



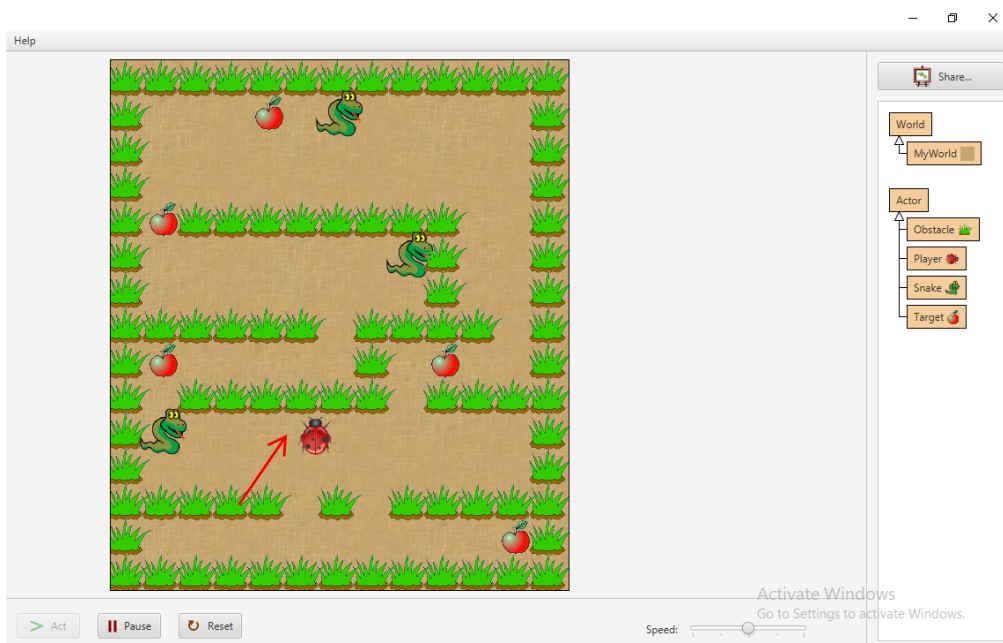
Pertama tambahkan perintah pada class Player dibagian “public void movePlayer” seperti yang diberi garis merah



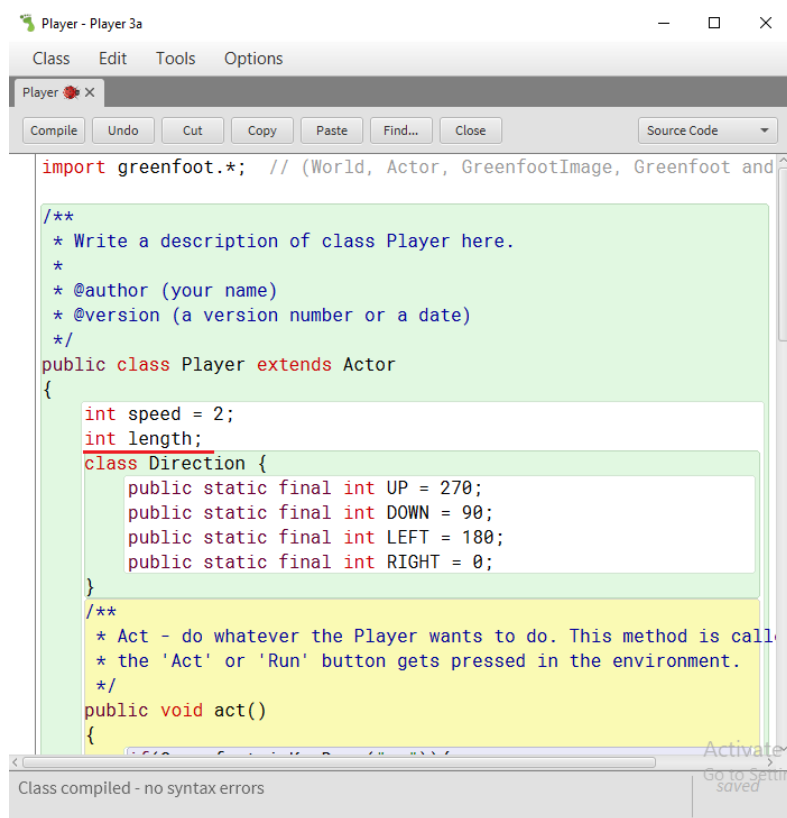
Makah hasilnya akan terlihat seperti gambar, sekarang aktor player tidak dapat berjalan melewati obstacle yang ada di dalam permainan



Tetapi belum selesai, disini kita akan sedikit merapikan agar aktor player menyentuh obstacle hanya bagian sisi luar saja, tidak seperti gambar diatas dimana player menyentuh obstacle hingga titik tengahnya walaupun sudah tidak dapat melewatinya, maka hasilnya akan Nampak seperti gambar dibawah ini



Open editor lagi pada class player, dan tambahkan variable “int length” seperti yang di beri tanda merah

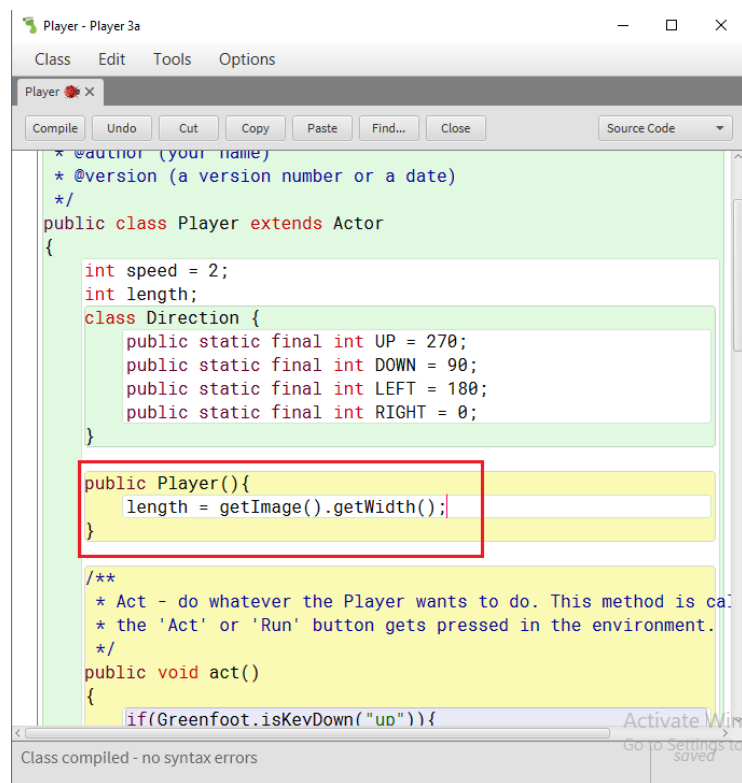


```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and others)

/**
 * Write a description of class Player here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Player extends Actor
{
    int speed = 2;
    int length;
    class Direction {
        public static final int UP = 270;
        public static final int DOWN = 90;
        public static final int LEFT = 180;
        public static final int RIGHT = 0;
    }

    /**
     * Act - do whatever the Player wants to do. This method is called when
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
    }
}
```

Selanjut nya tambahkan skrip seperti pada gambar

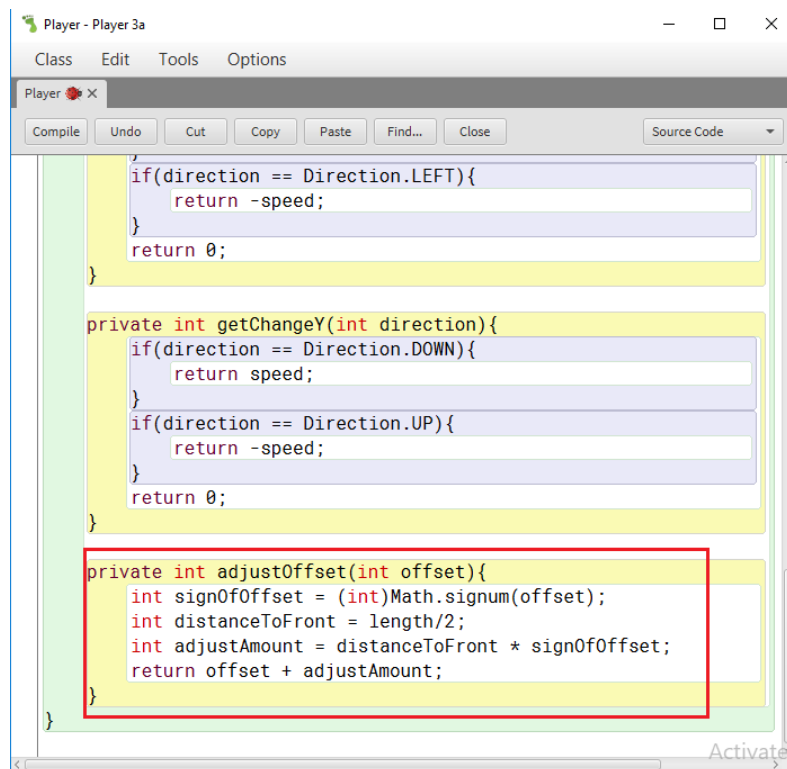


```
/**
 * @author (your name)
 * @version (a version number or a date)
 */
public class Player extends Actor
{
    int speed = 2;
    int length;
    class Direction {
        public static final int UP = 270;
        public static final int DOWN = 90;
        public static final int LEFT = 180;
        public static final int RIGHT = 0;
    }

    public Player(){
        length = getImage().getWidth();
    }

    /**
     * Act - do whatever the Player wants to do. This method is called when
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        if(Greenfoot.isKeyDown("up")){
    }
}
```

Kemudian tambahkan skrip seperti dibawah ini



```
Player - Player 3a
Class Edit Tools Options

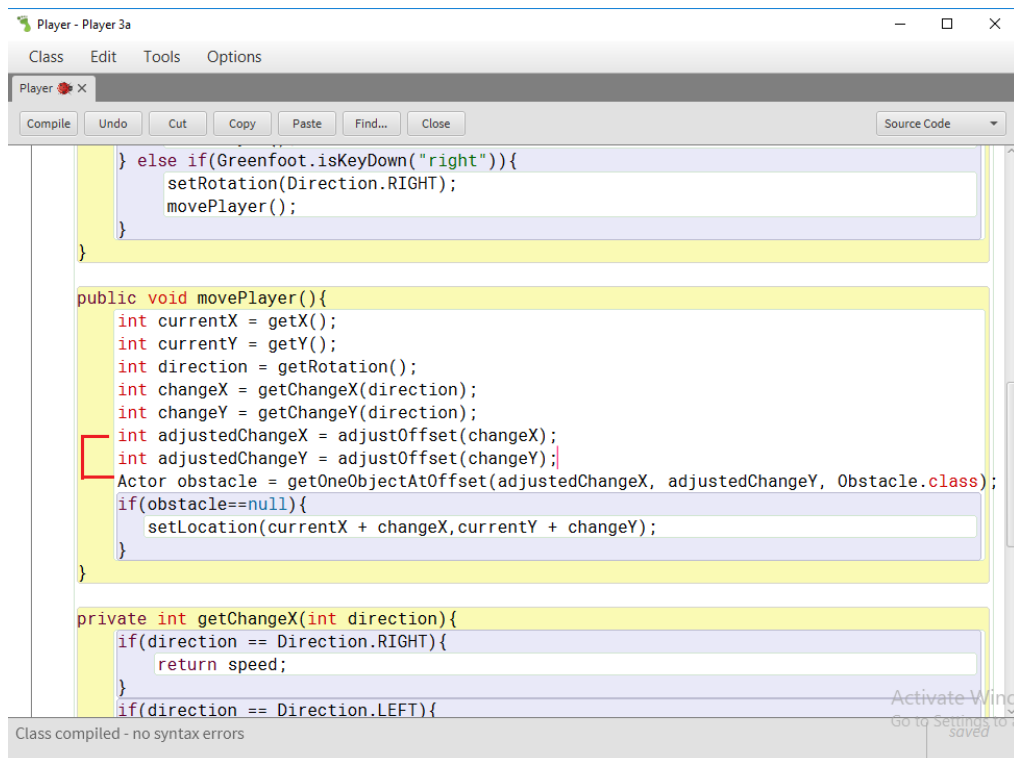
Player X
Compile Undo Cut Copy Paste Find... Close Source Code

if(direction == Direction.LEFT){
    return -speed;
}
return 0;

private int getChangeY(int direction){
    if(direction == Direction.DOWN){
        return speed;
    }
    if(direction == Direction.UP){
        return -speed;
    }
    return 0;
}

private int adjustOffset(int offset){
    int signOfOffset = (int)Math.signum(offset);
    int distanceToFront = length/2;
    int adjustAmount = distanceToFront * signOfOffset;
    return offset + adjustAmount;
}
```

Selanjutnya tambahkan skrip dibawah pada “public void movePlayer” dan ubah didalam kurung pada “Actor obstacle” yang awalnya hanya (changeX, changeY) kali ini tambahkan “adjusted” seperti yang terlihat pada gambar dibawah, silahkan Compile dan pastikan tidak ada syntax yang error dan lihat hasilnya



```
Player - Player 3a
Class Edit Tools Options

Player X
Compile Undo Cut Copy Paste Find... Close Source Code

} else if(Greenfoot.isKeyDown("right")){
    setRotation(Direction.RIGHT);
    movePlayer();
}

public void movePlayer(){
    int currentX = getX();
    int currentY = getY();
    int direction = getRotation();
    int changeX = getChangeX(direction);
    int changeY = getChangeY(direction);
    int adjustedChangeX = adjustOffset(changeX);
    int adjustedChangeY = adjustOffset(changeY);
    Actor obstacle = getOneObjectAtOffset(adjustedChangeX, adjustedChangeY, Obstacle.class);
    if(obstacle==null){
        setLocation(currentX + changeX,currentY + changeY);
    }
}

private int getChangeX(int direction){
    if(direction == Direction.RIGHT){
        return speed;
    }
    if(direction == Direction.LEFT){
        return -speed;
    }
    return 0;
}
```

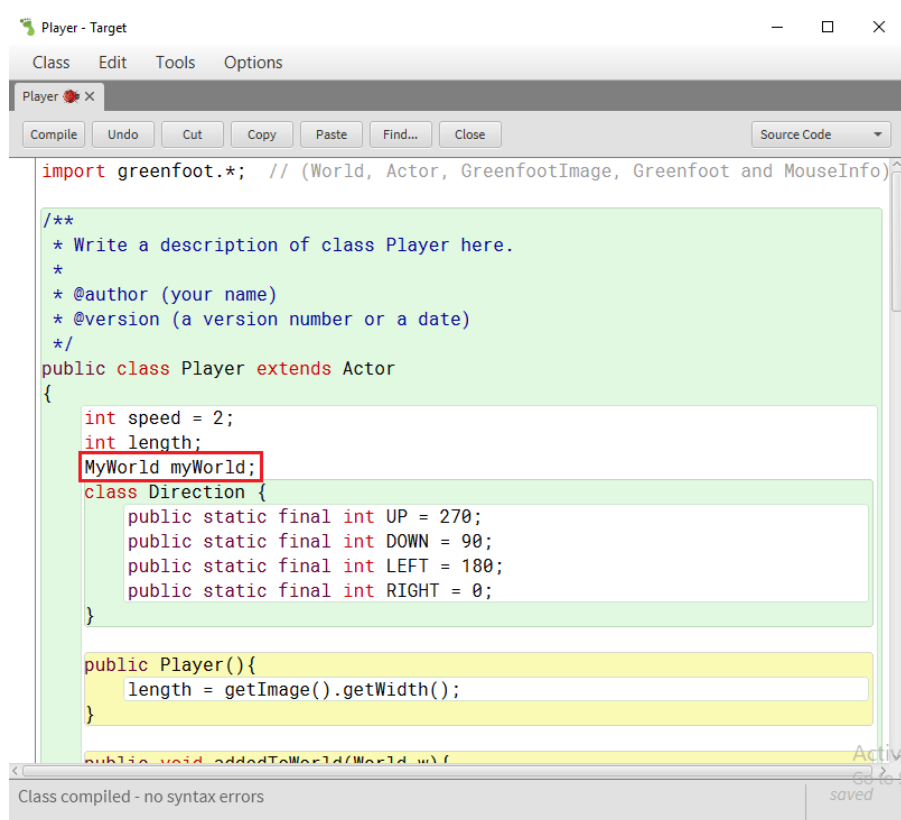
Class compiled - no syntax errors

5. Target

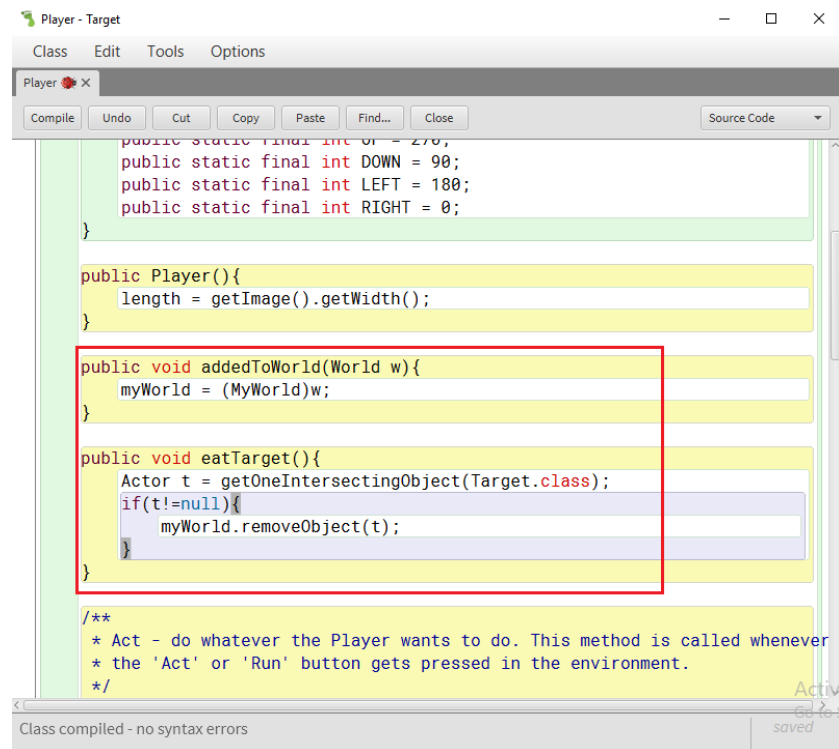
Dilangkah kelima ini kita akan membuat aktor player dapat memakan apel/target yang ada dalam permainan



Buatkan variable memanggil class "MyWorld" didalam class Player

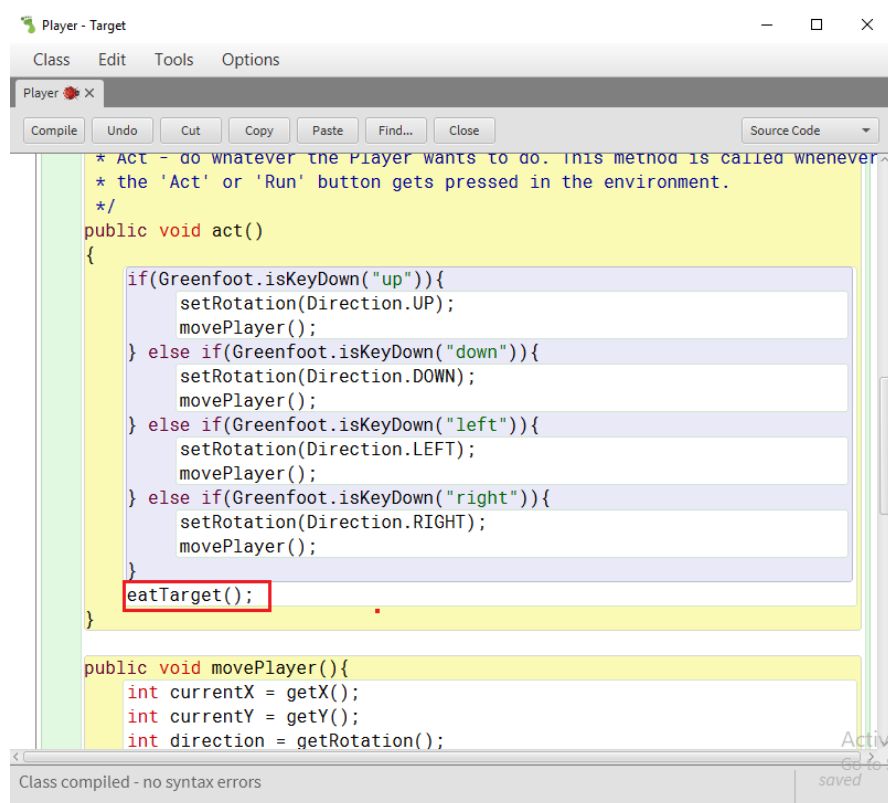


Lanjut dua perintah dibawah ini dimana apabila aktor Player menyentuh Target, maka target akan otomatis hilang dari permainan atau dimakan oleh aktor Player



```
Player - Target
Class Edit Tools Options
Player X
Compile Undo Cut Copy Paste Find... Close Source Code
public static final int UP = 270;
public static final int DOWN = 90;
public static final int LEFT = 180;
public static final int RIGHT = 0;
}
public Player(){
    length = getImage().getWidth();
}
public void addToWorld(World w){
    myWorld = (MyWorld)w;
}
public void eatTarget(){
    Actor t = getOneIntersectingObject(Target.class);
    if(t!=null){
        myWorld.removeObject(t);
    }
}
/**
 * Act - do whatever the Player wants to do. This method is called whenever
 * the 'Act' or 'Run' button gets pressed in the environment.
 */
Class compiled - no syntax errors saved
```

Tambahkan perintah dibawah ini pada “public void act”, kemudian compile dan lihat hasilnya apakah aktor player dapat memakan target berupa apel



```
Player - Target
Class Edit Tools Options
Player X
Compile Undo Cut Copy Paste Find... Close Source Code
* Act - do whatever the Player wants to do. This method is called whenever
* the 'Act' or 'Run' button gets pressed in the environment.
*/
public void act()
{
    if(Greenfoot.isKeyDown("up")){
        setRotation(Direction.UP);
        movePlayer();
    } else if(Greenfoot.isKeyDown("down")){
        setRotation(Direction.DOWN);
        movePlayer();
    } else if(Greenfoot.isKeyDown("left")){
        setRotation(Direction.LEFT);
        movePlayer();
    } else if(Greenfoot.isKeyDown("right")){
        setRotation(Direction.RIGHT);
        movePlayer();
    }
    eatTarget();
}
public void movePlayer(){
    int currentX = getX();
    int currentY = getY();
    int direction = getRotation();
}
Class compiled - no syntax errors saved
```

Jika berhasil maka akan seperti gambar dibawah ini



6. Snake

Dilangkah ini kita akan membuat aktor Snake bergerak secara otomatis, jadi aktor snake akan berjalan secara otomatis dan random ke segala arah, mulai dengan perintah yang bergaris merah

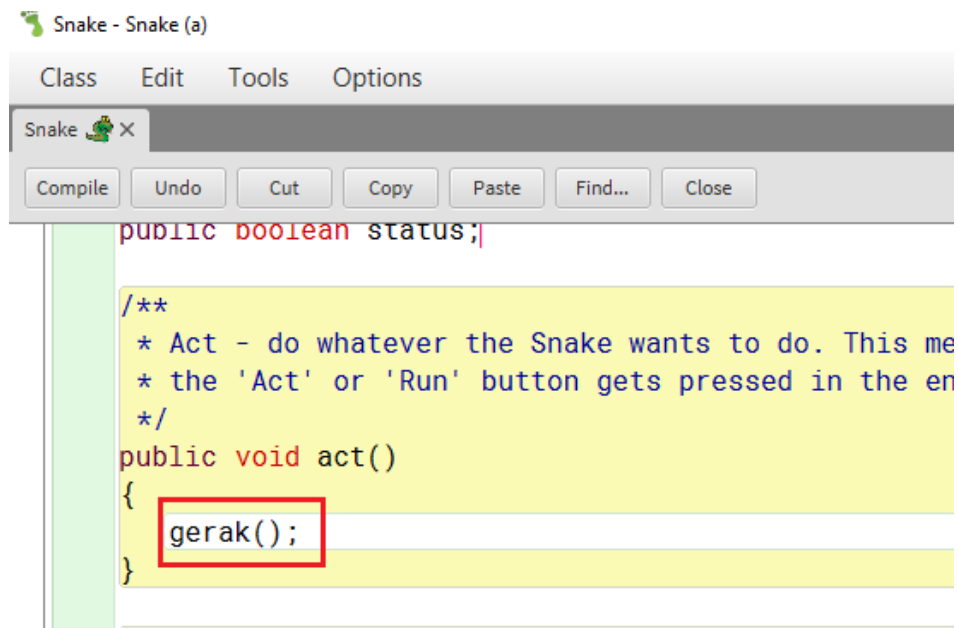
```
Snake - Snake (a)
Class Edit Tools Options
Snake X
Compile Undo Cut Copy Paste Find... Close Source Code
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Snake here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Snake extends Actor
{
    private static final int KANAN = 0;
    private static final int KIRI = 1;
    private static final int ATAS = 2;
    private static final int BAWAH = 3;
    public int ARAH=KANAN;
    public int GRK;
    public boolean status;

    /**
     * Act - do whatever the Snake wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        gerak();
    }
}
```

Class compiled - no syntax errors saved

Disini kita akan buat variable gerak

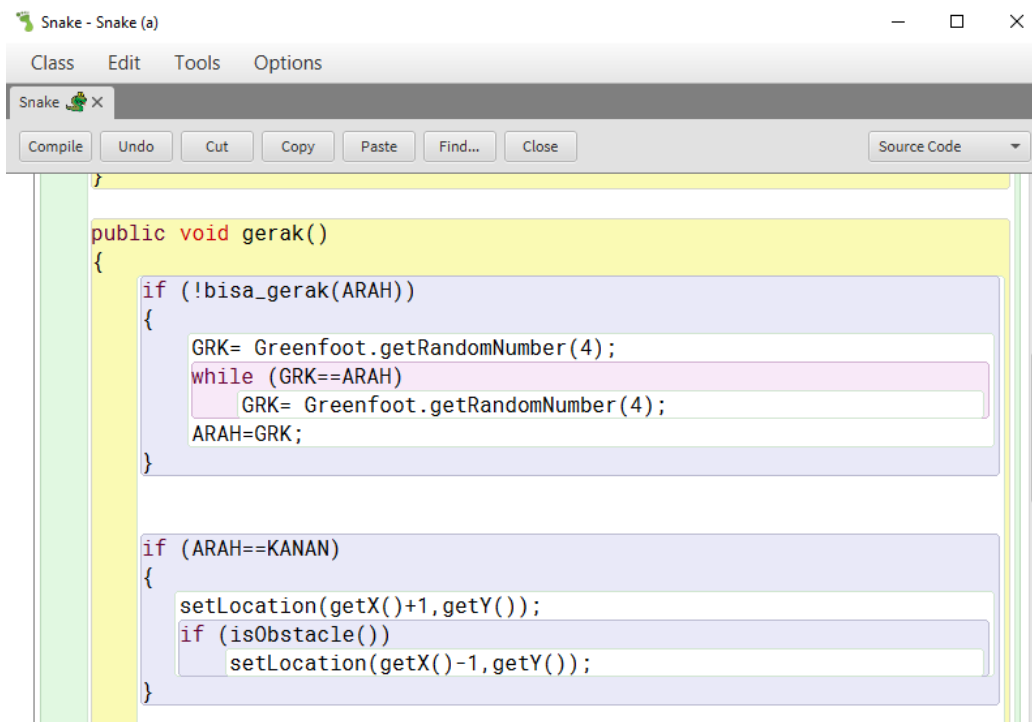


```
Snake - Snake (a)
Class Edit Tools Options
Snake
Compile Undo Cut Copy Paste Find... Close

public boolean status;

/**
 * Act - do whatever the Snake wants to do. This me
 * the 'Act' or 'Run' button gets pressed in the en
 */
public void act()
{
    gerak();
}
```

Setelah variable nya dibuat lanjut untuk memberikan perintah agar objek dapat bergerak random otomatis



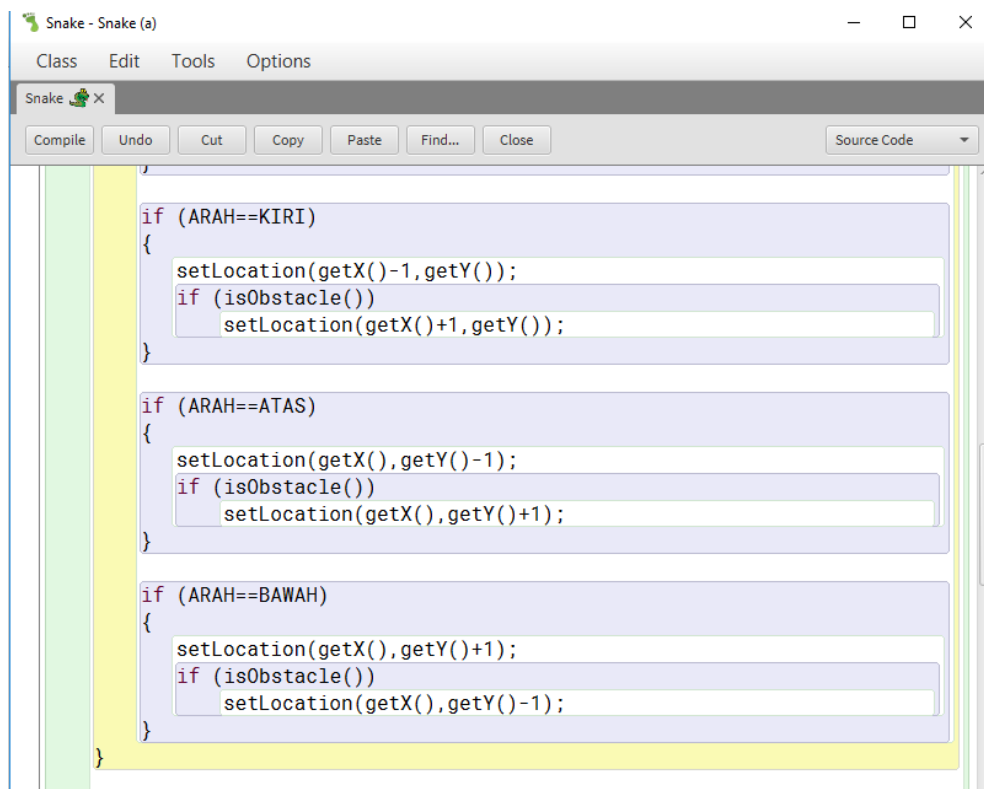
```
Snake - Snake (a)
Class Edit Tools Options
Snake
Compile Undo Cut Copy Paste Find... Close Source Code

}

public void gerak()
{
    if (!bisa_gerak(ARAH))
    {
        GRK= Greenfoot.getRandomNumber(4);
        while (GRK==ARAH)
        {
            GRK= Greenfoot.getRandomNumber(4);
        }
        ARAH=GRK;
    }

    if (ARAH==KANAN)
    {
        setLocation(getX()+1,getY());
        if (isObstacle())
        {
            setLocation(getX()-1,getY());
        }
    }
}
```

Lanjut dari skrip diatas



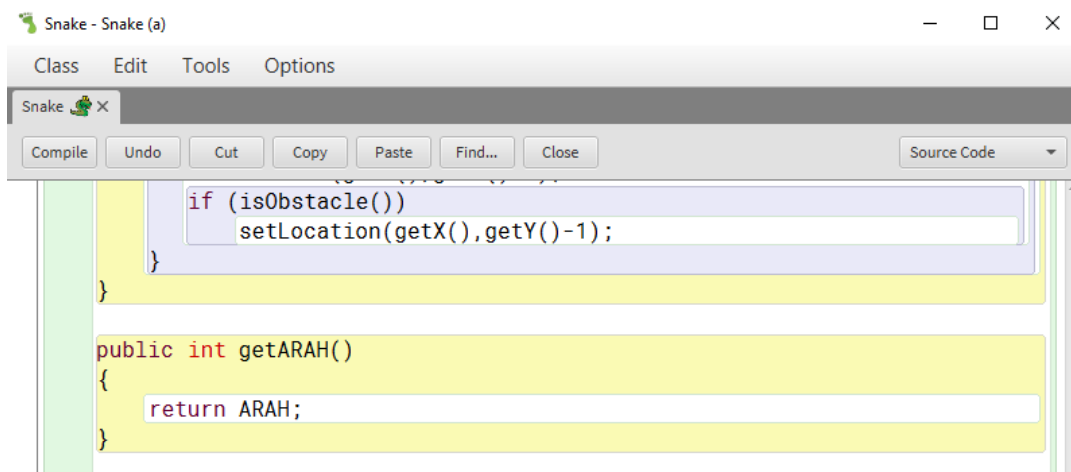
The screenshot shows a code editor window titled "Snake - Snake (a)". The code defines movement logic for a snake based on its direction (KIRI, ATAS, BAWAH). Each direction has a corresponding if-statement that calls setLocation() and checks for obstacles using isObstacle().

```
if (ARAH==KIRI)
{
    setLocation(getX()-1,getY());
    if (isObstacle())
        setLocation(getX()+1,getY());
}

if (ARAH==ATAS)
{
    setLocation(getX(),getY()-1);
    if (isObstacle())
        setLocation(getX(),getY()+1);
}

if (ARAH==BAWAH)
{
    setLocation(getX(),getY()+1);
    if (isObstacle())
        setLocation(getX(),getY()-1);
}
```

Selanjutnya kita akan memberikan arah pada objek Snake, pertama buatkan variable Arah



The screenshot shows a code editor window titled "Snake - Snake (a)". The code defines a public method getARAH() that returns the current direction of the snake (ARAH).

```
if (isObstacle())
{
    setLocation(getX(),getY()-1);
}

public int getARAH()
{
    return ARAH;
}
```

Dan lanjut skrip dibawah ini selain untuk arah disini kita akan membuat objek snake bergerak hanya didalam arena bermain objek snake tidak dapat melewati obstacle yang ada



```
public boolean bisa_gerak(int ARAH_GRK)
{
    Actor Obstacle;
    switch (ARAH_GRK)
    {
        case KANAN : Obstacle = getOneObjectAtOffset(1,0,Obstacle.class);
                     if (Obstacle!=null || getX()==9) status=false; else status=true; break;

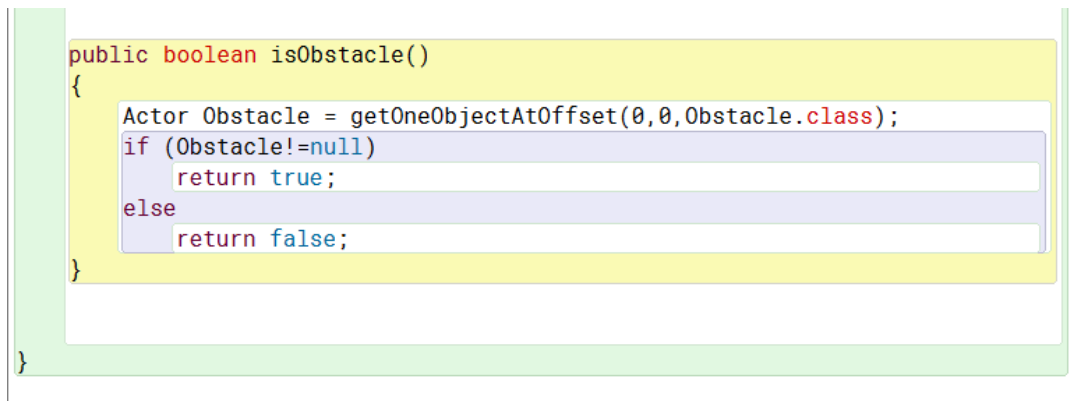
        case KIRI  : Obstacle = getOneObjectAtOffset(-1,0,Obstacle.class);
                     if (Obstacle!=null || getX()==0) status=false; else status=true; break;

        case ATAS   : Obstacle = getOneObjectAtOffset(0,-1,Obstacle.class);
                     if (Obstacle!=null || getY()==0) status=false; else status=true; break;

        case BAWAH  : Obstacle = getOneObjectAtOffset(0,1,Obstacle.class);
                     if (Obstacle!=null || getY()==6) status=false; else status=true; break;

        default     : status=false;
    }
    return (status);
}
```

Langkah terakhir tambahkan skrip dibawah ini



```
public boolean isObstacle()
{
    Actor Obstacle = getOneObjectAtOffset(0,0,Obstacle.class);
    if (Obstacle!=null)
        return true;
    else
        return false;
}
```

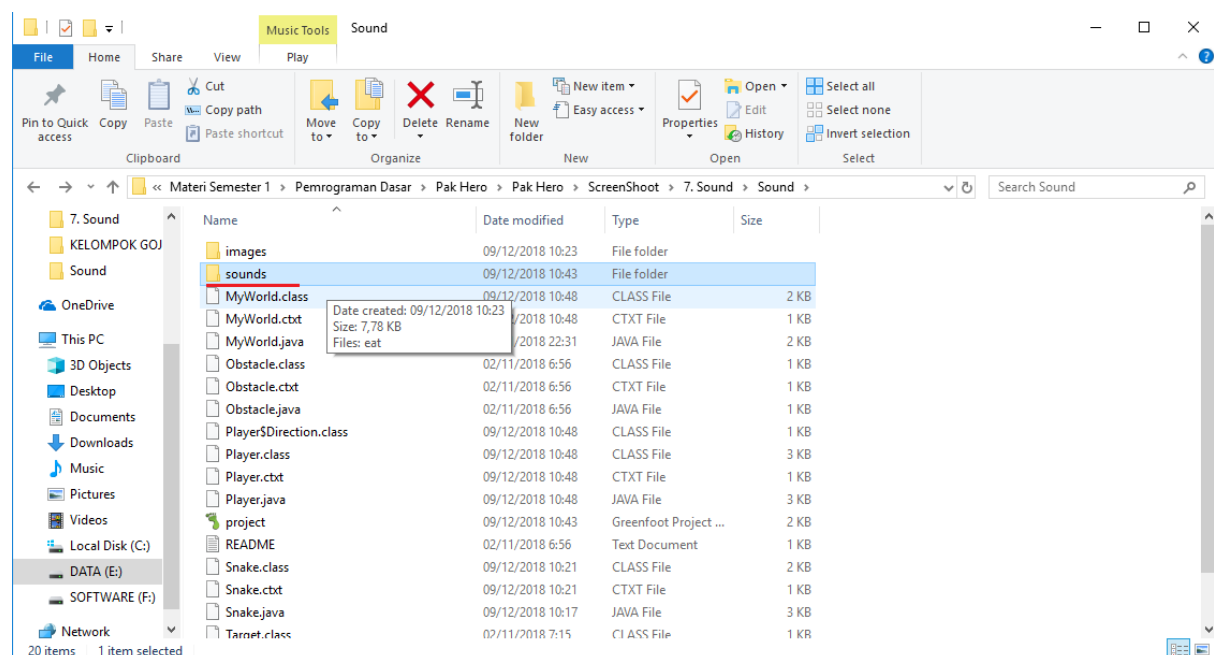

Setelah berhasil di compile dan coba di jalankan, maka akan terlihat seperti gambar di bawah ini



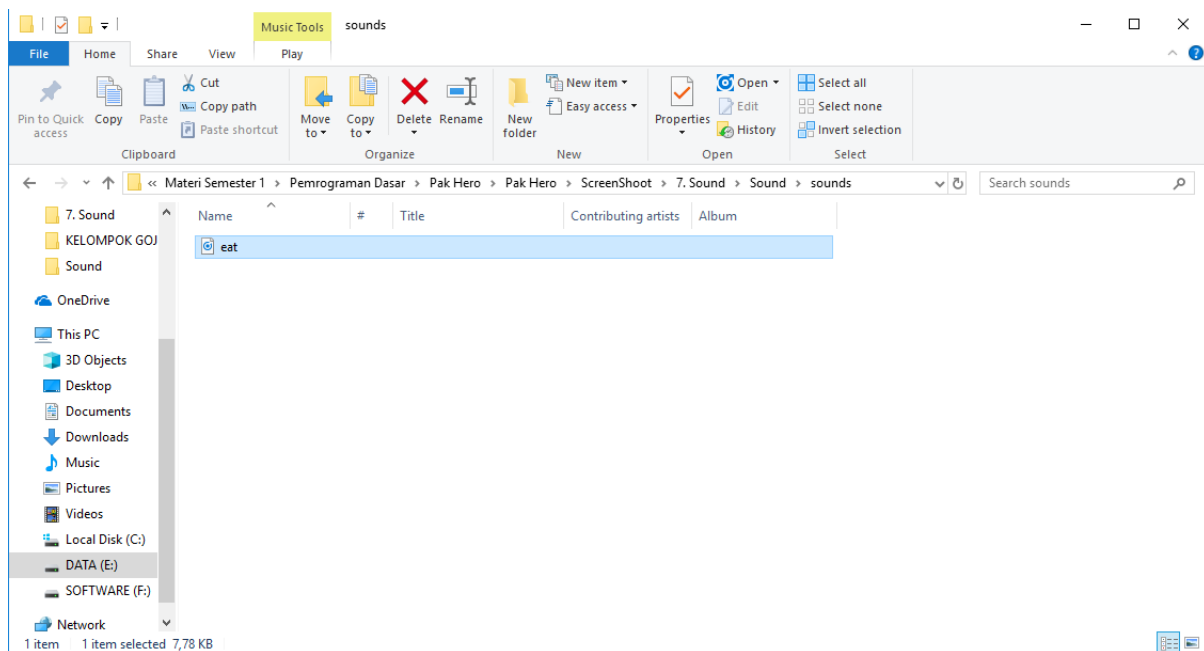
7. Sound

Langkah ketujuh kita akan memberikan efek sound pada game yang akan dibuat, ada beberapa sound yang akan diberikan, yang pertama sound pada saat Player memakan Apel, kedua pada saat Player menyentuh Snake dan yang ketiga efek sound untuk permainan.

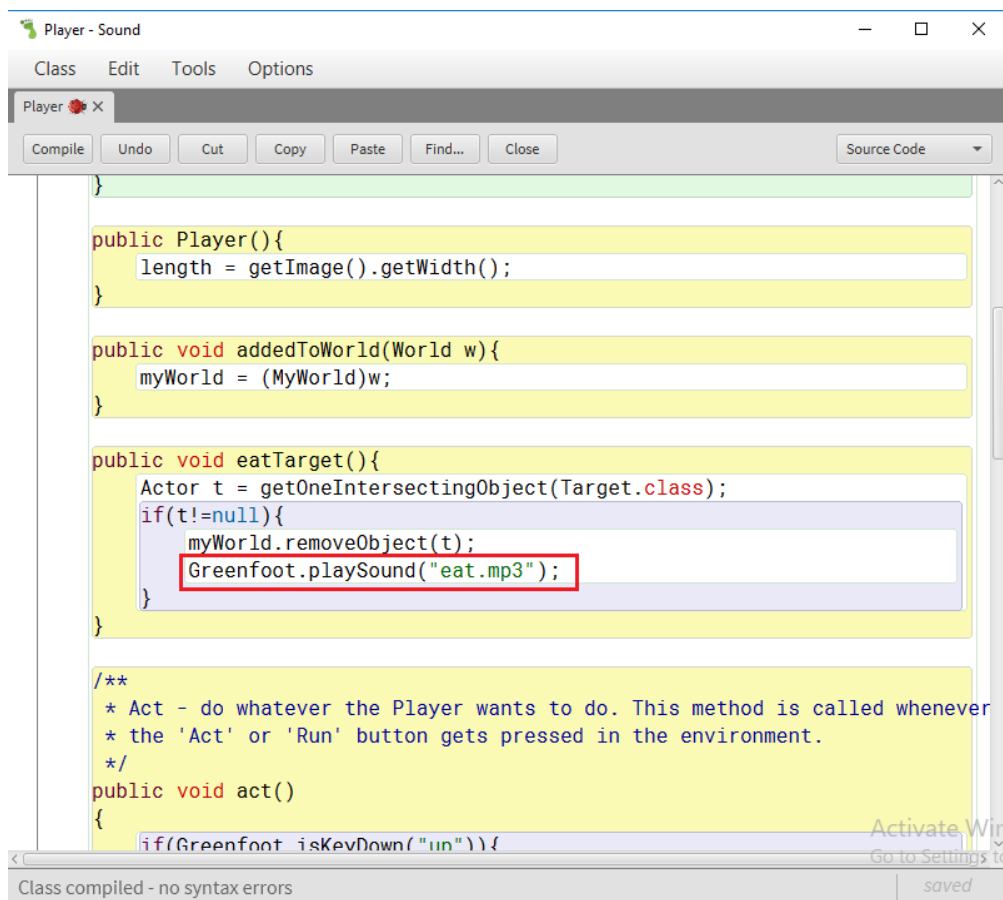
Pertama kita akan membuat efek sound pada saat player memakan buah, caranya siapkan sound yang terbaik dan pindahkan sound tersebut kedalam folder “sounds” pada greenfoot



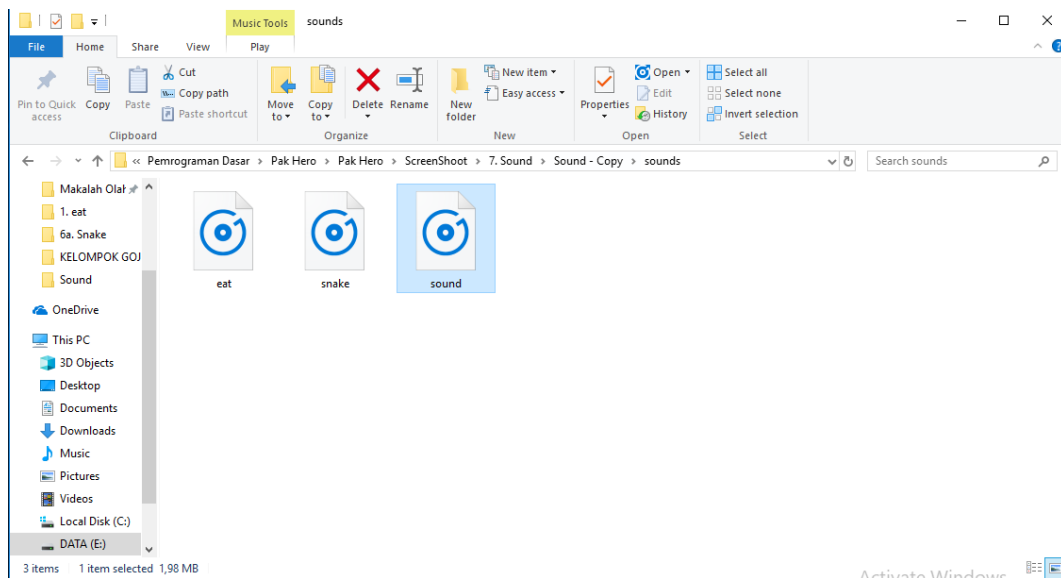
Seperti dibawah ini, pastikan format file tersebut format musik seperti contoh mp.3



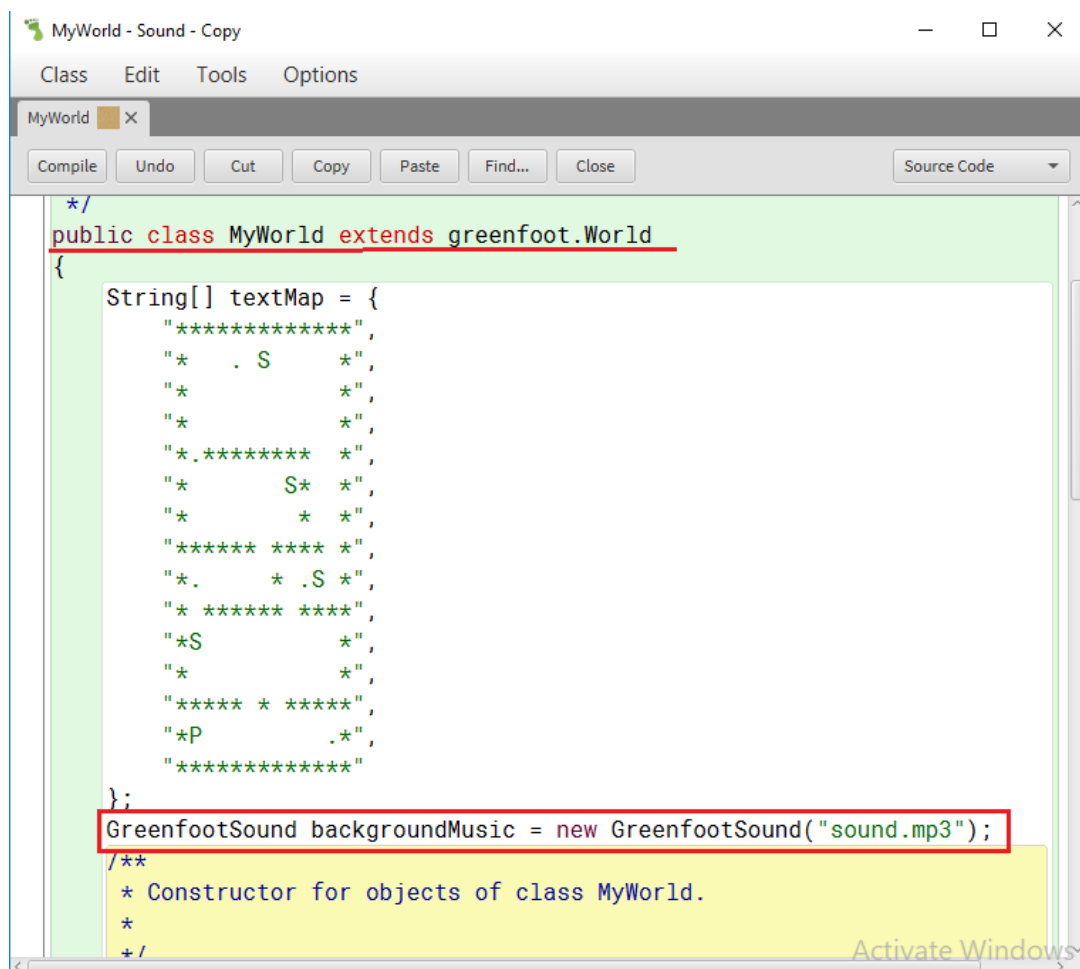
Setelah itu Open Editor pada actor Player kemudian tambahkan skrip seperti yang digaris merah pastikan nama file dan formatnya sesuai



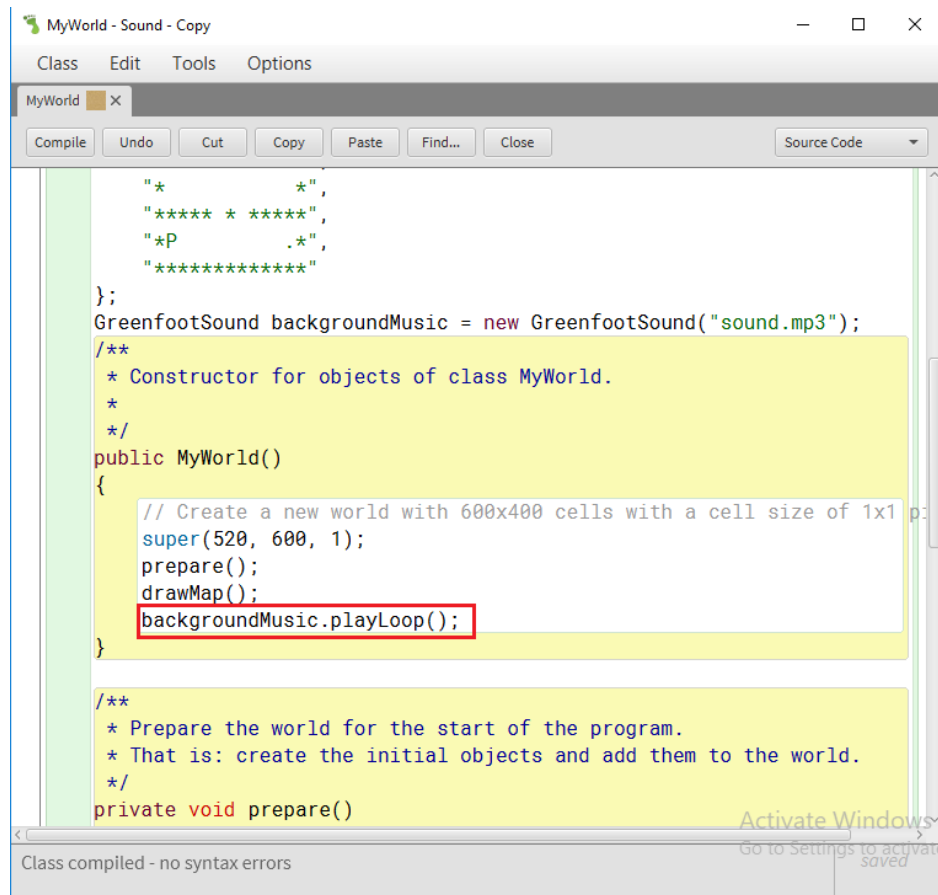
Selanjutnya kita akan memberikan musik pada latar permainan, langkah pertama sama seperti sebelumnya yaitu pindahkan file music yang kita inginkan ke dalam folder “Sounds” pada greenfoot



Open Editor pada class “MyWorld” lalu tambahkan skrip seperti dalam kotak merah yang berada didalam “public class MyWorld”



Tambahkan skrip didalam “public MyWorld” dibawah ini untuk menjalankan efek sound, setelah itu compile dan jalankan



```
MyWorld - Sound - Copy
Class Edit Tools Options
MyWorld x
Compile Undo Cut Copy Paste Find... Close Source Code
/*
 *
 */
***** * *****
 *P .*,
 *****

};
GreenfootSound backgroundMusic = new GreenfootSound("sound.mp3");
/**
 * Constructor for objects of class MyWorld.
 *
 */
public MyWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 p:
    super(520, 600, 1);
    prepare();
    drawMap();
    backgroundMusic.playLoop();
}

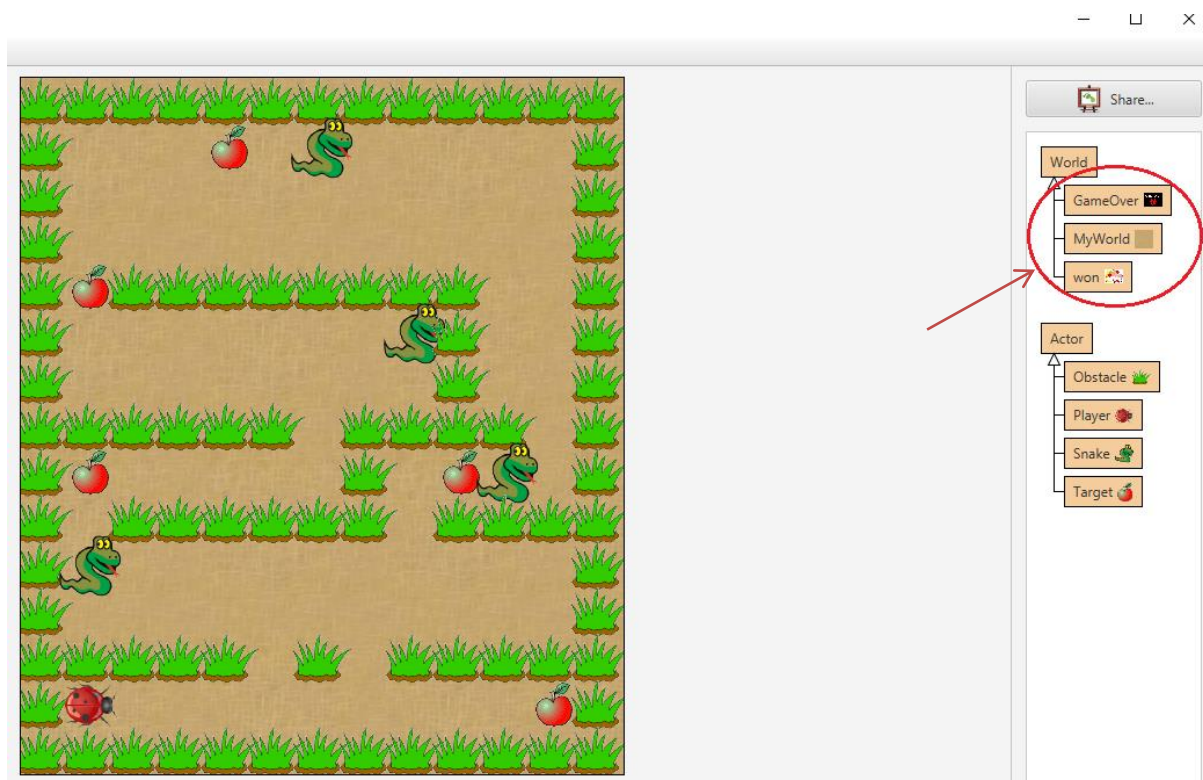
/**
 * Prepare the world for the start of the program.
 * That is: create the initial objects and add them to the world.
 */
private void prepare()
```

Class compiled - no syntax errors

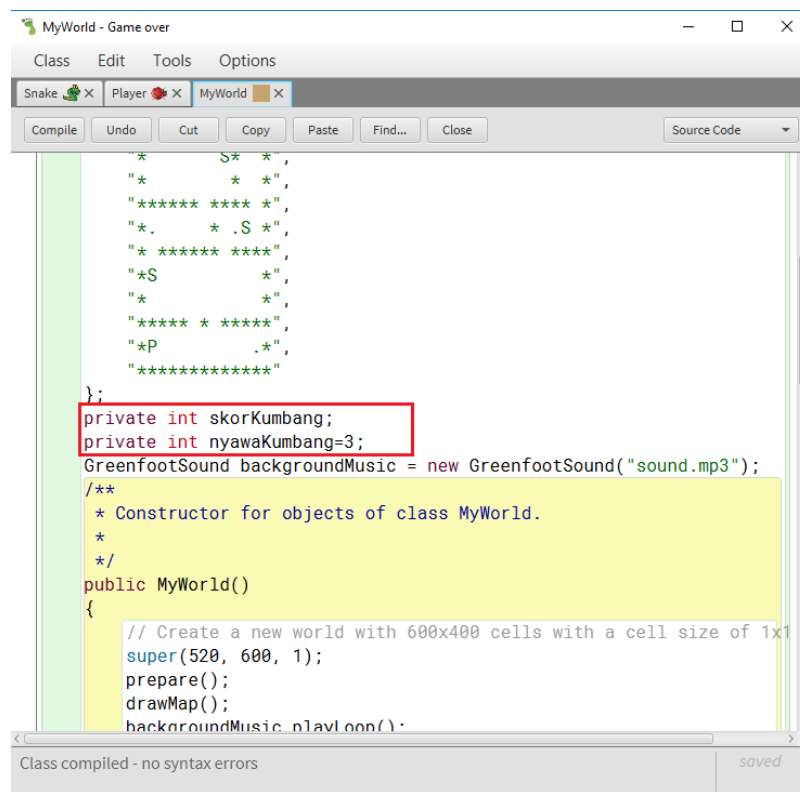
8. Clear Score and Game Over

Tahap terakhir kita akan membuat Score dimana apabila Player telah mencapai score yang ditetapkan maka permainan akan berakhir dan anda menang, sebaliknya apabila player belum mencapai score dan nyawa yang diberikan telah habis maka permainan berakhir dengan Game Over

Langkah pertama kita tambahkan class world, disini saya menambahkan dua class world yaitu class won dan class GameOver, dimana class world tersebut telah di tambahkan gambar untuk won dan gameover



Selanjutnya masuk langkah yang menyenangkan yaitu coding, Open Editor pada class MyWorld dan tambahkan variable Skor dan nyawa kumbang/player, disini kita dapat menentukan berapa banyak nyawa yang akan diberikan pada player, saya menambahkan sebanyak tiga nyawa



```

class MyWorld {
    Snake s;
    Player p;
    Map m;
    GreenfootSound backgroundMusic = new GreenfootSound("sound.mp3");

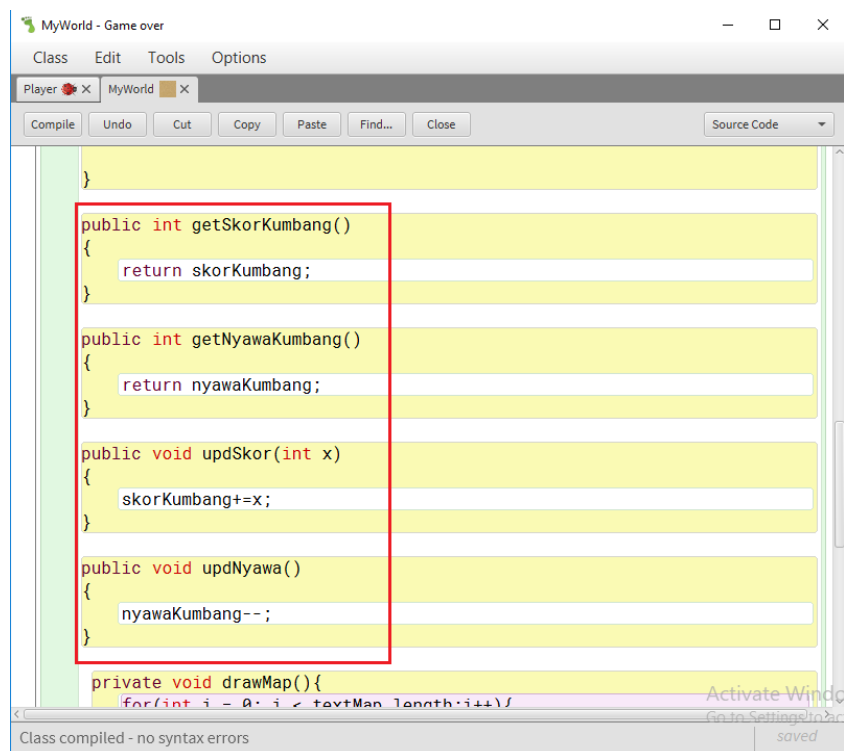
    /**
     * Constructor for objects of class MyWorld.
     */
    public MyWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1
        super(520, 600, 1);
        prepare();
        drawMap();
        backgroundMusic.playLoop();
    }

    private int skorKumbang;
    private int nyawaKumbang=3;

    // ... (other methods) ...
}

```

Lanjut skrip dibawah ini, untuk skor karena angkanya akan bertambah jadi kita tambahkan tanda (+) dan untu nyawa kita tambahkan tanda (-) agar nyawanya bisa berkurang



```

class MyWorld {
    // ... (previous code) ...

    public int getSkorKumbang()
    {
        return skorKumbang;
    }

    public int getNyawaKumbang()
    {
        return nyawaKumbang;
    }

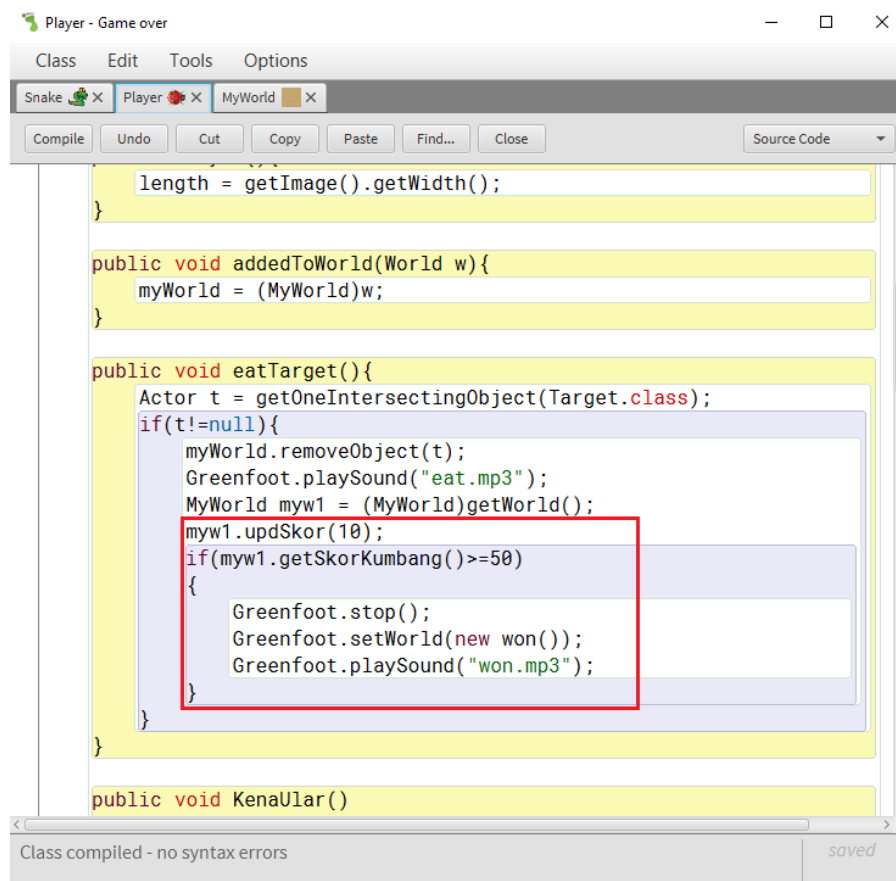
    public void updSkor(int x)
    {
        skorKumbang+=x;
    }

    public void updNyawa()
    {
        nyawaKumbang--;
    }

    private void drawMap(){
        for(int i = 0; i < textMap.length;i++){
            // ... (drawing code) ...
        }
    }
}

```

Sekarang OpenEditor pada class Player, kita akan menambahkan skrip seperti didalam kotak merah yang terletak di “public void eatTarget” dimana apabila player memakan target maka score akan bertambah 10 begitu pun seterusnya sampai score mencapai 50 maka permainan akan berhenti dan tampilan world akan berubah seperti gambar yang ada di world won dan disini saya menambahkan sound untuk kemenangan



```
Player - Game over
Class Edit Tools Options
Snake X Player X MyWorld X
Compile Undo Cut Copy Paste Find... Close Source Code

length = getImage().getWidth();
}

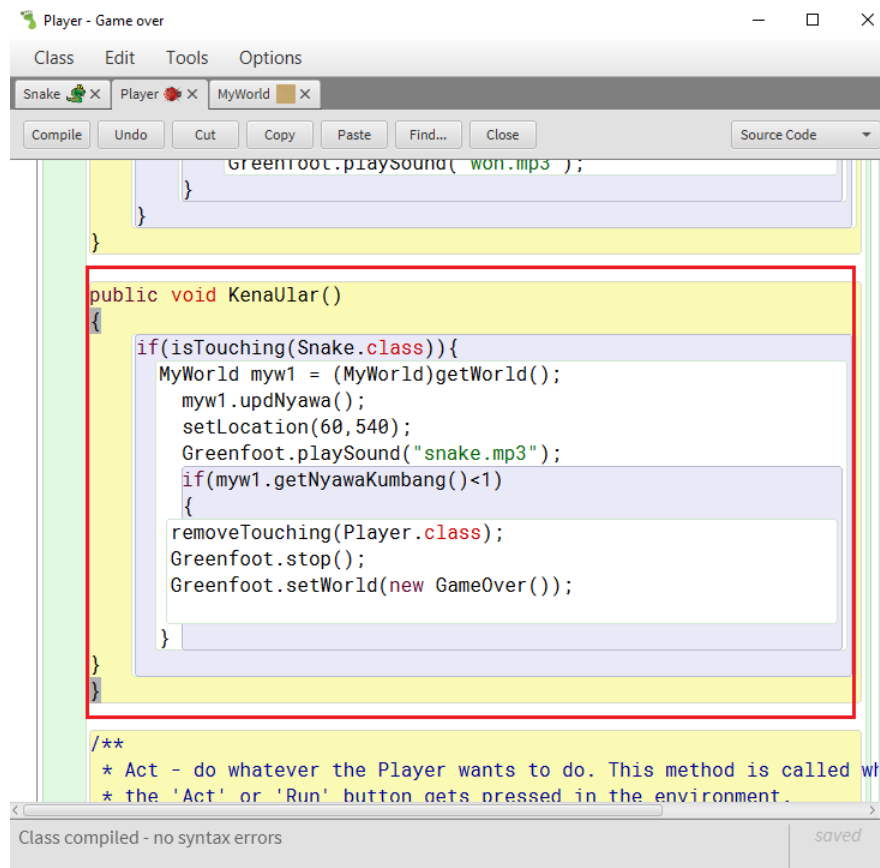
public void addToWorld(World w){
    myWorld = (MyWorld)w;
}

public void eatTarget(){
    Actor t = getOneIntersectingObject(Target.class);
    if(t!=null){
        myWorld.removeObject(t);
        Greenfoot.playSound("eat.mp3");
        MyWorld myw1 = (MyWorld)getWorld();
        myw1.updSkor(10);
        if(myw1.getSkorKumbang()>=50)
        {
            Greenfoot.stop();
            Greenfoot.setWorld(new won());
            Greenfoot.playSound("won.mp3");
        }
    }
}

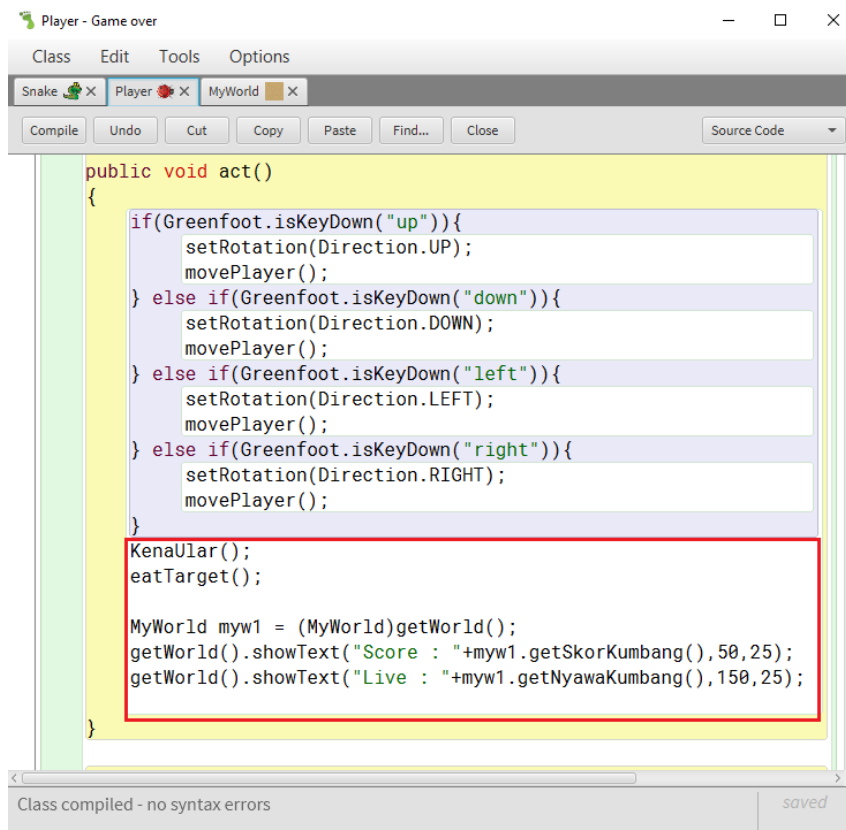
public void KenaUlar()
```

Class compiled - no syntax errors saved

Selanjutnya kita akan membuat GameOver, skripnya hampir sama seperti kita membuat won, bedanya disini apabila Player menyentuh Snake maka akan ada efek sound dan nyawa player berkurang satu apabila player menyentuh snake sebanyak tiga kali maka permainan berakhir dengan GameOver dan world akan berganti dan menampilkan world pada class world GameOver



Terakhir tampilkan Score dan nyawa yang telah dibuat sesuai keinginan anda dengan menentukan titik koordinat nya, caranya tambahkan skrip di dalam kotak merah pada “public void act”, silahkan compile dan pastikan tidak ada yang error kemudian jalankan permainan



```

public void act()
{
    if(Greenfoot.isKeyDown("up")){
        setRotation(Direction.UP);
        movePlayer();
    } else if(Greenfoot.isKeyDown("down")){
        setRotation(Direction.DOWN);
        movePlayer();
    } else if(Greenfoot.isKeyDown("left")){
        setRotation(Direction.LEFT);
        movePlayer();
    } else if(Greenfoot.isKeyDown("right")){
        setRotation(Direction.RIGHT);
        movePlayer();
    }

    KenaUlar();
    eatTarget();

    MyWorld myw1 = (MyWorld)getWorld();
    getWorld().showText("Score : "+myw1.getSkorKumbang(),50,25);
    getWorld().showText("Live : "+myw1.getNyawaKumbang(),150,25);
}

```

Class compiled - no syntax errors

Jika berhasil skor dan nyawa akan tampil seperti pada gambar letak nya diatas



Ini adalah tampilan apabila Game Over



Dan ini tampilan apabila menang dengan efek sound



C. Cara Bermain



Jalankan Player menggunakan tombol arah pada keyboard untuk memakan buah apel yang ada dalam permainan, apabila apel dimakan maka skor akan bertambah sebanyak 10 poin.

Jangan sampai player menyentuh lawan yaitu ular, apabila player menyentuhnya maka nyawa player akan berkurang satu dan posisi player akan kembali ke posisi awal.

Tetapi apabila player dapat memakan semua apel yang ada maka permainan berakhir dengan kemenangan sebaliknya apabila player menyentuh ular sampai nyawanya habis maka permainan berakhir dengan GameOver.

Daftar Pustaka

<https://www.greenfoot.org>

<https://www.youtube.com/>

<https://www.google.com/> (Tutorial game greenfoot)