

Modul Pelatihan :

Pembuatan Game Animasi Menggunakan Greenfoot

Bab 1 PENDAHULUAN

Greenfoot adalah perangkat lunak yang didesain untuk pemula agar dapat terbiasa dengan Pemrograman Berorientasi Objek(Object-Oriented Programming), yang mendukung pengembangan aplikasi bergambar dengan memakai bahasa pemrograman Java™. Dengan greenfoot, belajar Bahasa pemrograman menjadi lebih mudah dan menyenangkan.

Langkah-langkah dalam pembuatan greenfoot akan dijelaskan dalam modul ini, akan lebih mudah jika mengikuti modul ini dengan mempraktekkannya sekaligus.

1.1 Unduh greenfoot

Untuk dapat menggunakan greenfoot, software yang dibutuhkan adalah :

1. Java

- a. JDK (Java Development Kit) : Untuk membangun program Java
- b. JRE (Java Runtime Environment) : Untuk menjalankan program Java

Dapat diunduh di: <http://www.oracle.com/technetwork/java/index.html>

- 1. Cari link download Java SE 7 atau Java SE 8
- 2. Klik Download JDK / JRE

2. Greenfoot

Dapat diunduh di : <http://www.greenfoot.org/download>

1.2 Install greenfoot

Sebelum menginstall greenfoot, anda diharuskan untuk menginstall java terlebih dahulu. Cara penginstallan java dan greenfoot cukup mudah yaitu dengan mengikuti segala instruksi yang ada saat proses penginstalan berlangsung.

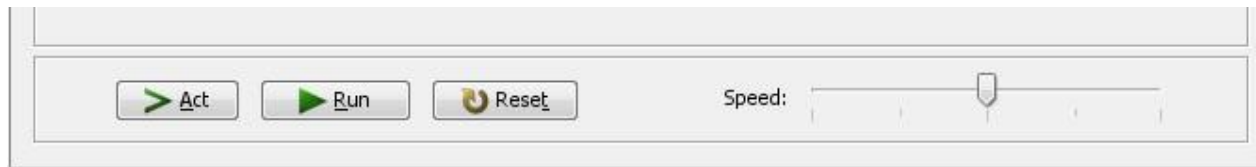
Bab 2 Pengenalan Greenfoot

2.1 Scenario

Scenario adalah sebuah permainan atau simulasi yang diimplementasikan di Greenfoot.

Untuk mengeksekusi suatu scenario, dapat menggunakan **control eksekusi** yang terdiri dari :

- **Act** : Menjalankan semua aksi di skenario satu kali
- **Run/pause** : Menjalankan semua aksi di skenario berulang-ulang sampai tombol Pause di-klik
- **Reset** : Memberhentikan sebentar skenario atau mengulang skenario kembali ke posisi awal ➤
- Speed** : Menjalankan aksi lebih cepat atau lambat



Gambar 1 Kontrol Eksekusi

2.2 Classes

Class memberitahukan skenario bagaimana bentuk dari obyek dan aksinya ketika skenario dijalankan.

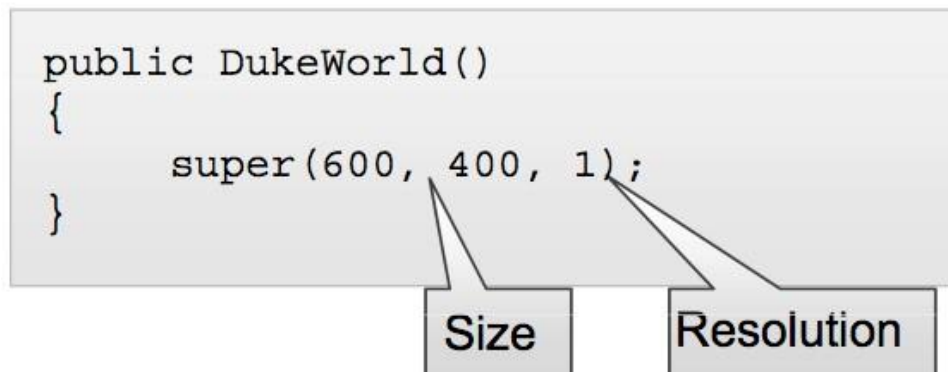
2.2.1 Superclass

Tipe dari Superclass di Greenfoot ➤

World :

- Memegang subclass yang menyediakan background gambar untuk skenario world
- Mendefinisikan ukuran dan resolusi world

Ukuran dan Resolusi World

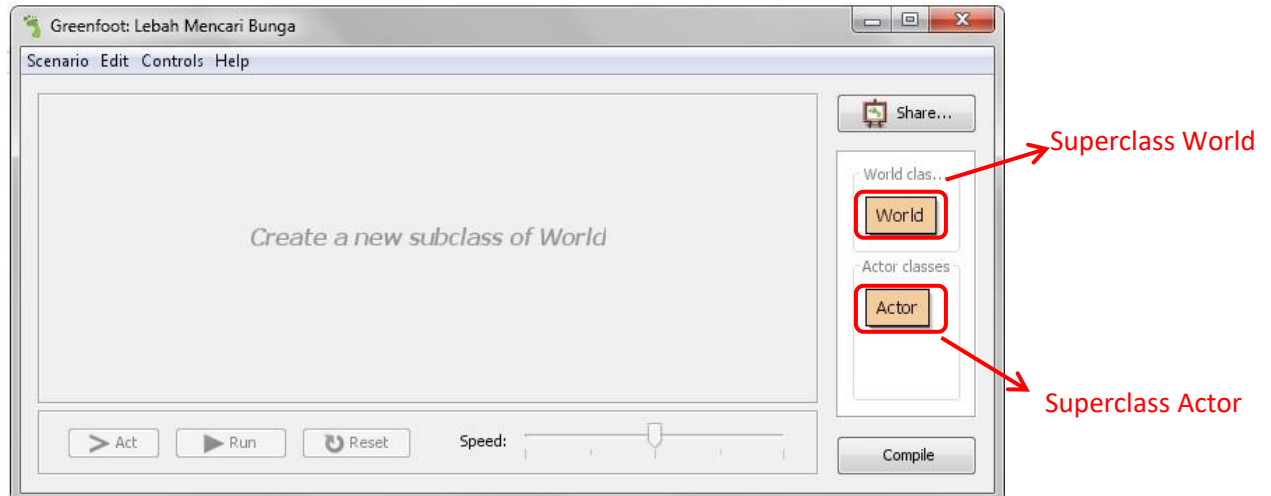


Gambar 2 merubah ukuran World

Kata kunci “super” menandakan bahwa constructor DukeWorld memanggil constructor dari superclass-nya yaitu constructor World(). #Class DukeWorld adalah subclass dari class World

➤ Actor

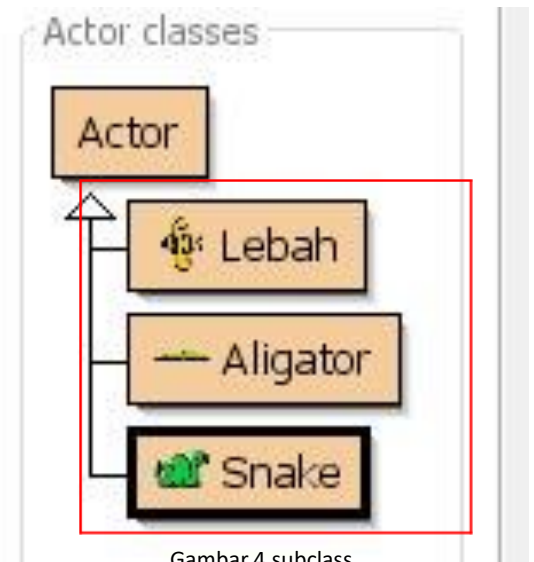
Memegang subclass-subclass yang menghasilkan instances yang beraksi di skenario



Gambar 3 superclass World dan Actor

2.2.2 Subclass Sebuah Class khusus

1. Menurunkan semua properti superclass Actor, contohnya mendefinisikan lebih awal aksi-aksi yang dapat dilakukan oleh Aktor subclass
2. Dapat memiliki properti baru yang dibuat programmer khusus untuk subclass seperti gambar dan aksi
3. Properti subclass
 - ❖ Subclass memiliki hubungan “is-a” terhadap Superclass
contoh : Lebah is a subclass of the Actor superclass
 - ❖ Properti dapat dimodifikasi (seperti nama Class, gambar yang ditampilkan, atau aksi-aksi yang dilakukan) .
 - ❖ Panah pada hirarki Class menunjukkan hubungan antara subclass dengan superclass



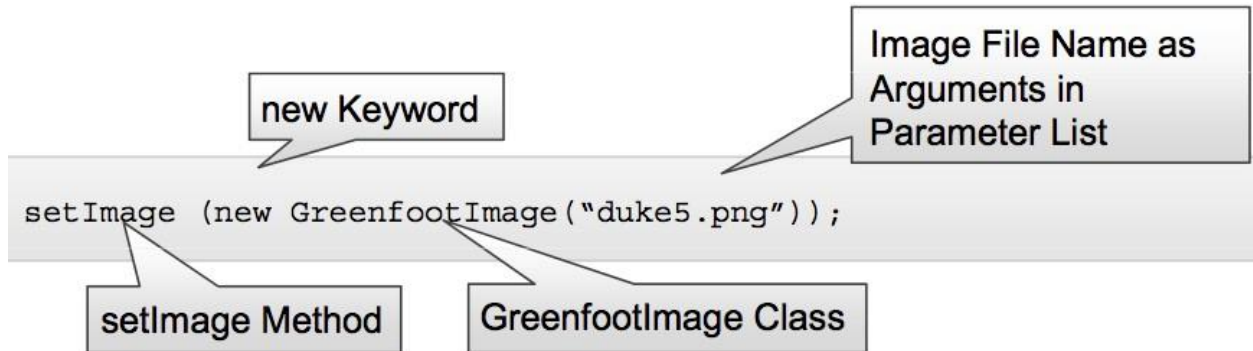
Gambar 4 subclass

2.2.3 Inheritance

Inheritance adalah setiap subclass mewarisi (inherits) method-methodnya dari superclass-nya.
Contoh : Sebuah Kelas Lebah akan mewarisi method-method superclassnya yaitu Kelas Actor, sehingga Lebah dikatakan sebagai subclass dari Kelas Actor

2.2.4 Class GreenfootImage

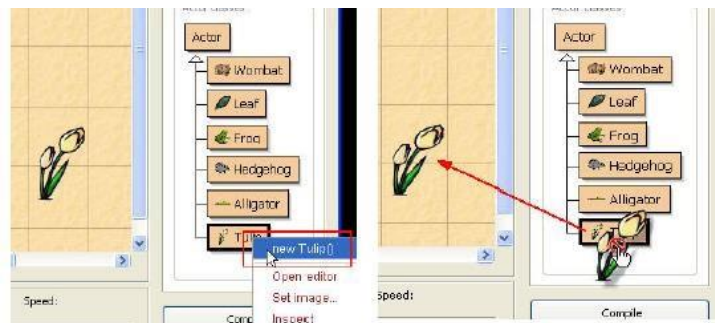
- Class GreenfootImage memungkinkan actor untuk menjaga gambar pada game tetap muncul menggunakan object bertipe GreenfootImage.
- Image yang digunakan harus sudah ada di folder images scenario game.
- Object GreenfootImage dapat digunakan sebagai parameter dalam method setImage.



Gambar 5 GreenfootImage Class

2.3 Object / Instances

Instances adalah obyek-obyek dari sebuah class yang beraksi pada sebuah skenario. Sebuah instances memiliki karakteristik Class, akan tetapi dapat dimanipulasi dan diubah.



Gambar 6 penambahan object

2.4 Method

Method adalah sebuah kumpulan operasi atau tugas dimana instance sebuah class dapat menjalankan. Ketika sebuah method dipanggil / di-invoke, method tersebut akan menjalankan operasi atau tugas yang dituliskan di source code.

2.4.1 Komponen Method

Sebuah method memiliki beberapa komponen yang mendeskripsikan operasi-operasi atau tugas-tugas yang dijalankan

- **Return type** : Menspesifikasikan data yang dikembalikan oleh method
- **Method name** : Mendeskripsikan pekerjaannya si method
- **Parameter List** : Informasi yang masuk kedalam panggilan method



Gambar 7 contoh method

2.4.2 Return Type

Return type adalah sebuah kata diawal method yang mengindikasikan tipe informasi yang akan diberikan oleh method ketika method tersebut dipanggil.

Dua tipe method :

- **Void** : Memberikan sebuah command kepada obyek
- **Non Void** : Menanyakan obyek sebuah pertanyaan



Gambar 8 contoh return type

2.4.2.1 Method Void Return Type

Method dengan tipe void menghasilkan sebuah command yang membawa sebuah aksi

- termasuk kata-kata “void”
- tidak mengembalikan informasi mengenai obyek diterbitkan untuk membuat obyek melakukan sesuatu

void Method akan dipanggil :

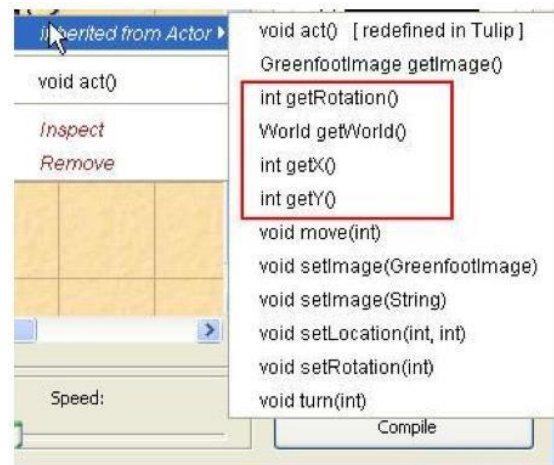
- Untuk menentukan posisi obyek pertama kali
- Memerintahkan obyek untuk menjalankan aksi-aksi pada permainan.



Gambar 9 Method dengan return type Void

2.4.2.2 Method Return Type non-void

- Method dengan return type non-void menanyakan obyek sebuah pertanyaan
- Method Signature tidak terdapat kata-kata “void”
- Method mengembalikan informasi mengenai obyek, tetapi tidak merubahnya dan menggerakkannya.
Contohnya :
 - Integer (ditampilkan sebagai int)
 - Merujuk ke number
 - Pertanyaan ke obyek : berapa banyak?
 - Boolean
 - nilai Returnnya true atau false
 - tipe pertanyaan yang akan ditanyakan ke obyek

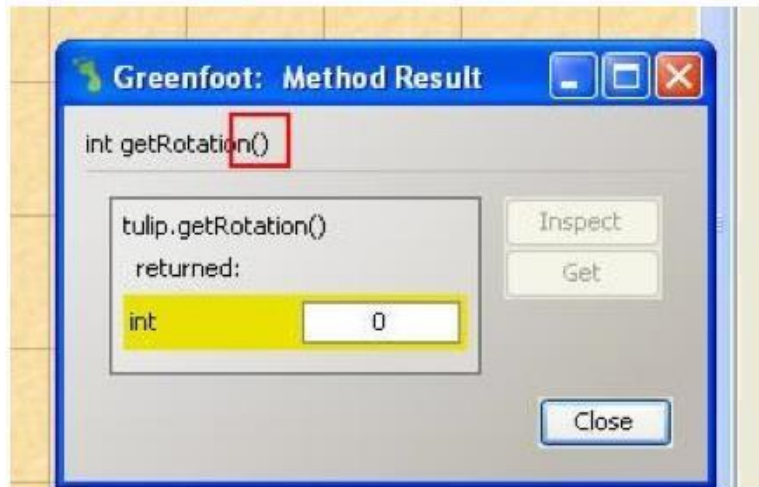


Gambar 10 contoh method dengan return type non-Void

2.4.2.3 Method Parameter Lists

□ Method Parameter Lists mengindikasikan apakah method memerlukan informasi tambahan ketika dipanggil and tipe informasi seperti apa □ Parameter Lists ditampilkan dengan tanda kurung □ Memiliki dua state :

- Empty : Tidak ada data yang diharapkan untuk memanggil method (get Rotation())
- Non-Empty : Memiliki data dan berharap satu atau lebih parameter untuk memanggil method (turn() method)



Gambar 11 Method Parameter list

2.4.3 Method Act

Method act: method yang mendeskripsikan hal-hal yang dilakukan oleh object saat pengguna meng-klik tombol Act atau Run.

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

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator 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.
    }
}
```

Gambar 12 Method Act

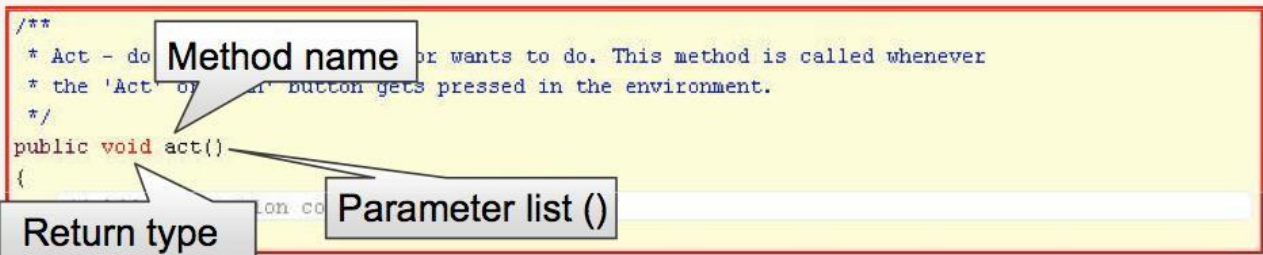
2.4.4 Method Signature

Method signature berisi:

- Return type
- Nama Method
- Daftar parameter

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

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do something the actor wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // on co
    }
}
```



Gambar 13 Method Signature

2.4.5 Method dari Class Actor

| Method Name | Description |
|--------------------------------|--|
| void move(int distance) | Assigns the object a number of steps to move, or the command to simply move when the Act or Run buttons are clicked. |
| void turn(int amount) | Assigns the object a number of degrees to turn. |
| void act() | Gives the object the opportunity to perform an action in the scenario. Method calls are inserted into this method. |
| void setLocation(int x, int y) | Assigns a new location for this object. |
| void setRotation(int rotation) | Sets a new rotation for this object. |

Gambar 14 Method dari class Actor

2.4.6 Method dengan mengembalikan orientasi object

| Method Name | Description |
|-------------------|--|
| int getRotation() | Returns the current rotation of the object. |
| World getWorld() | Returns the name of the world the object is in. |
| int getX() | Returns the x-coordinate of the object's current location. |
| int getY() | Returns the y-coordinate of the object's current location. |

Gambar 15 method dengan mengembalikan orientasi object



Gambar 16 Menampilkan orientasi object

2.4.7 Notasi Dot

Memanggil method dari class lain dapat menggunakan notasi dot.

Format notasi dot mencakup:

- Nama class/object dimana method yang ingin dipanggil berada
- Dot (titik)
- Nama method yang dipanggil
- Parameter
- Titik koma untuk mengakhiri statement

```
class-name.method-name (parameters);  
object-name.method-name (parameters);
```

Gambar 17 format notasi dot

2.4.8 Method addObject

Method addObject didefinisikan di class World Digunakan untuk membuat object di class tertentu Terdiri dari 3 parameter:

- Object. Kata kunci “new” digunakan untuk membuat object baru.
- Koordinat x
- Koordinat y

Method signature dari method addObject:

```
void addObject(Actor object, int x, int y)
```

Gambar 18 membuat method addObject

Pemanggilan method addObject dilengkapi dengan 3 parameter:

- Object. Kata kunci “new” digunakan untuk membuat object baru.
- Koordinat x
- Koordinat y

```
new Constructor-name()
```

Gambar 19 pemanggilan method addObject

2.4.9 Method atWorldEdge

Method-method yang digunakan pada definisi method atWorldEdge:

- getX : mengembalikan koordinat X dari Actor
- getY : mengembalikan koordinat Y dari Actor
- getWorld : mengembalikan world dari Actor
- getHeight : mengembalikan lebar/tinggi dari image
- getWidth : mengembalikan panjang dari image

2.4.10 Menggunakan Kontrol Keyboard

Di Greenfoot, kontrol dengan keyboard:

- Berada di class Greenfoot
- Merupakan method static (method isKeyDown)
- Mengembalikan boolean (true atau false)
- Menggunakan String sebagai parameternya
- Dapat digunakan sebagai kondisi dalam IF

```
public static boolean isKeyDown(String key)
```

Gambar 20 method kontrol keyboard

2.4.11 Pemanggilan Method

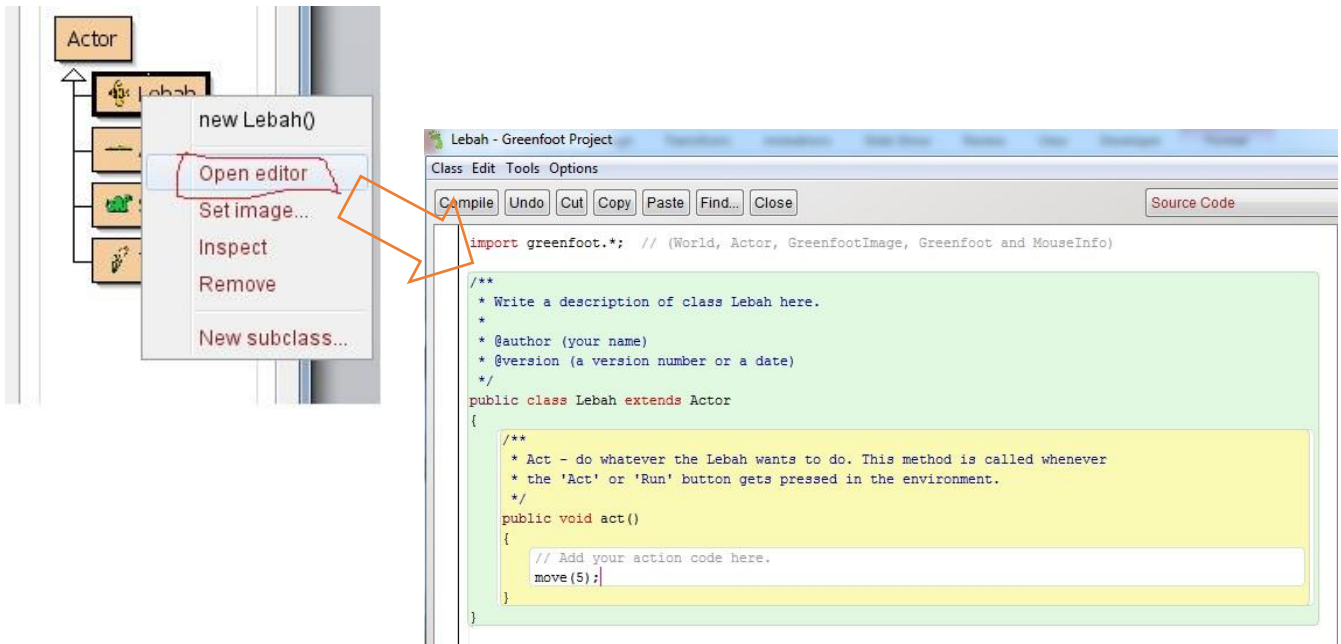
Pemanggilan method mencakup:

- **Nama method**
- **Parameter** yang sesuai daftar parameter
- **Titik koma (;)** di akhir statement

Hasil pemanggilan method dapat disimpan dalam sebuah **variabel** jika return type dari method adalah non-void.

2.5 Source Code

- Source code Java adalah blueprint yang mendefinisikan bagaimana program Anda berfungsi.
- Source code Java berisi perintah-perintah kepada object untuk beraksi dan berinteraksi.
- Source code dikelola di Code Editor (klik kanan di nama class, lalu klik "Open editor").
- Fungsi dari code editor: menulis/memodifikasi aksi dari instance class, me-review method dan properti yang diwarisi class, me-review method yang dibuat khusus untuk class tersebut.



Gambar 21 Source code

2.5.1 Komponen Source code

2.5.1.1 Class Description

Class Description: kumpulan komentar yang mendeskripsikan class, biasanya mencakup: Apa yang dilakukan class, Nama pembuat class, Versi atau tanggal terakhir dimodifikasi

```

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

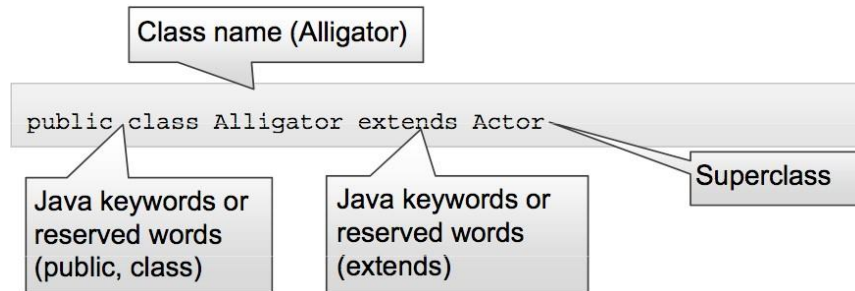
/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}

```

Gambar 22 class description

2.5.1.1 Class Definition

- Class Definition mencakup:
- Kata kunci “**class**” untuk mendefinisikan class
 - Kata kunci “**extends**” untuk mendefinisikan pewarisan
 - Nama class (subclass)
 - Nama superclass yang diwarisi subclass



Gambar 23 Class Definition

2.5.1.2 Mendefinisikan Class Isi dari sebuah class:

- **Variabel (atau field)** : menyimpan data dalam object/instance
- Sebuah variabel memiliki syntax penulisan sebagai berikut:
 - # Access modifier: *public/protected/default/private*
 - # Tipe data yang disimpan
 - # Nama variabel Contoh :

```

private GreenfootImage imagel;
private GreenfootImage image2;

```

Access modifier

Tipe data

Nama variabel

- **Constructor** : method khusus yang dieksekusi otomatis saat instance baru dibuat.
Karakteristik Constructor:
 - Mendefinisikan properti object, misalnya ukuran dan resolusi world
 - Tidak memiliki return type (bedakan dengan method yang wajib ada return type void atau non-void)
 - Nama constructor sama dengan nama class
- **Method** : menyediakan perilaku/aksi dari object/instance

2.5.1.3 Komentar ○ **Komentar** : berisi penjelasan source code, ○ Komentar tidak dieksekusi ○ Komentar ditandai dengan // atau /* */

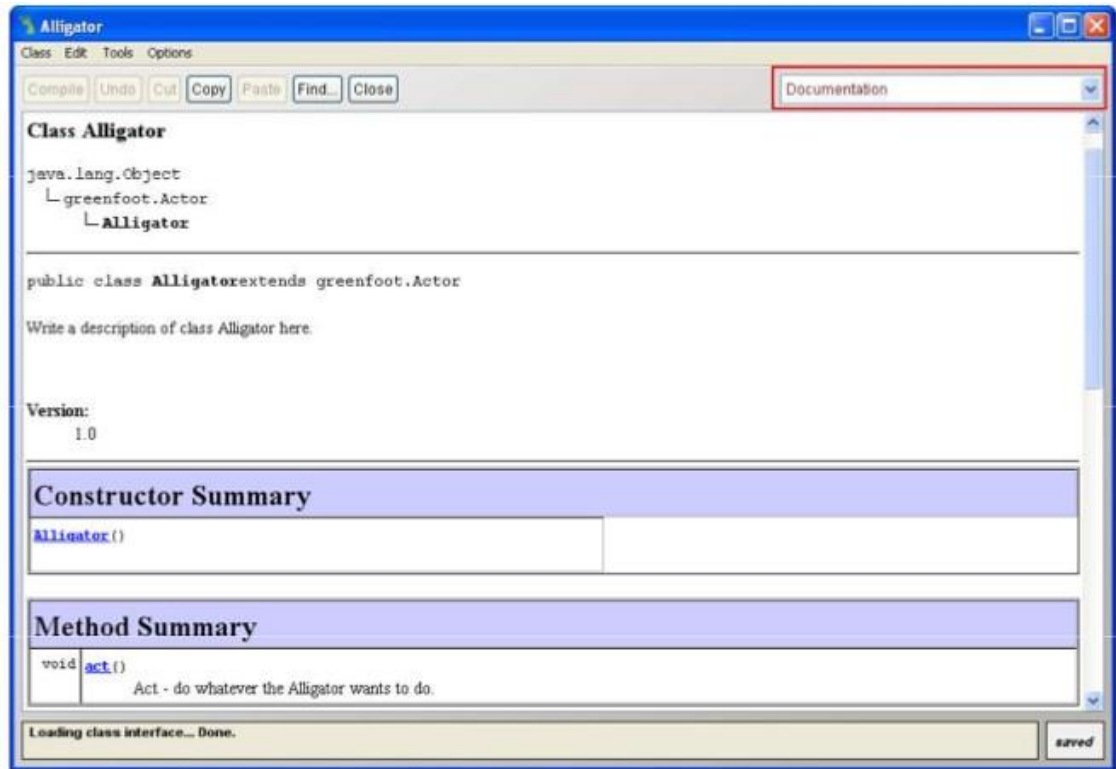
```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Alligator here.
 *
 * @author: Oracle Academy
 * @version 1.0
 */
public class Alligator extends Actor
{
    /**
     * Act - do whatever the Alligator 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.
    }
}
```

Gambar 24 Komentar

2.5.1.4 Dokumentasi Class

- Setiap class dapat dibuat dokumentasi dalam bentuk html.
- Untuk melihat dokumentasi, klik View, lalu klik Documentation.



Gambar 25 Documentasi Class

2.6 Operator

2.6.1 Operator logika

| Logic Operator | Means | Definition |
|-----------------------|-------|---|
| Exclamation Mark (!) | NOT | Reverses the value of a boolean expression (if b is true, !b is false. If b is false, !b is true). |
| Double ampersand (&&) | AND | Combines two boolean values, and returns a boolean value which is true if and only if both of its operands are true. |
| Two lines () | OR | Combines two boolean variables or expressions and returns a result that is true if either or both of its operands are true. |

Gambar 26 operator logika

2.6.2 Operator Penugasan (Assignment)

- Operator penugasan dapat digunakan untuk menyimpan nilai/objek dalam sebuah variabel.
- Ketika object di-assign ke dalam suatu variabel, maka variabel akan berisi reference ke object tersebut.
- Operator penugasan adalah “=”
- Syntax: `Variable = expression;`
- Contoh implementasi:

```
image1 = new GreenfootImage("duke.png");  
image2 = new GreenfootImage("duke2.png");
```

Gambar 27 operator penugasan

2.6.3 Operator ==

- Operator == digunakan untuk memeriksa apakah dua buah nilai sama, mengembalikan boolean (true/false)
- Operator == berbeda dengan operator = :
 - Operator == adalah operator perbandingan
 - Operator = adalah operator penugasan

2.6.4 Operator perbandingan

Gunakan operator <, <=, >, >=, ==, != untuk membandingkan 2 buah bilangan bulat/desimal.

```
if (Greenfoot.getRandomNumber(100) < 6)  
{  
    turn(Greenfoot.getRandomNumber(20));  
}
```

Gambar 28 operator perbandingan

2.6.5 IF-ELSE

- Jika condition bernilai true, maka statement di dalam body IF akan dieksekusi.
- Jika condition bernilai false, maka statement di dalam body ELSE akan dieksekusi.

```
if (condition)  
{  
    statements;  
}  
else  
{  
    statements;  
}
```

Gambar 29 if-else



Gambar 30 kondisi if-else

2.7 Debugging

Debugging adalah sebuah proses menemukan dan menghapus bugs atau error pada sebuah program komputer.

```
public class Tulip extends Actor
{
    /**
     * Act - do whatever the Tulip 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.
        move(5)
    }
}
```

Gambar 31 Syntax Error Example

```
public void act()
{
    // Add your action code here.
    move(5)
}
```

}; expected

2.8 Greenfoot API Interface

Di Greenfoot: klik menu 'Help' > klik 'Greenfoot Class Documentation'



Gambar 32 API

2.9 Mengakhiri Mengakhiri Scenario Game

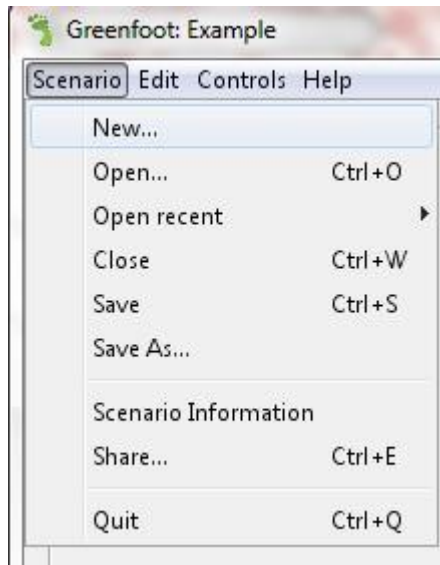
- Untuk mengakhiri scenario, dapat menggunakan method stop(). Game tidak dapat dimainkan kembali
- Implementasi penggunaan method stop():

```
Greenfoot.stop();
```

Gambar 33 method stop

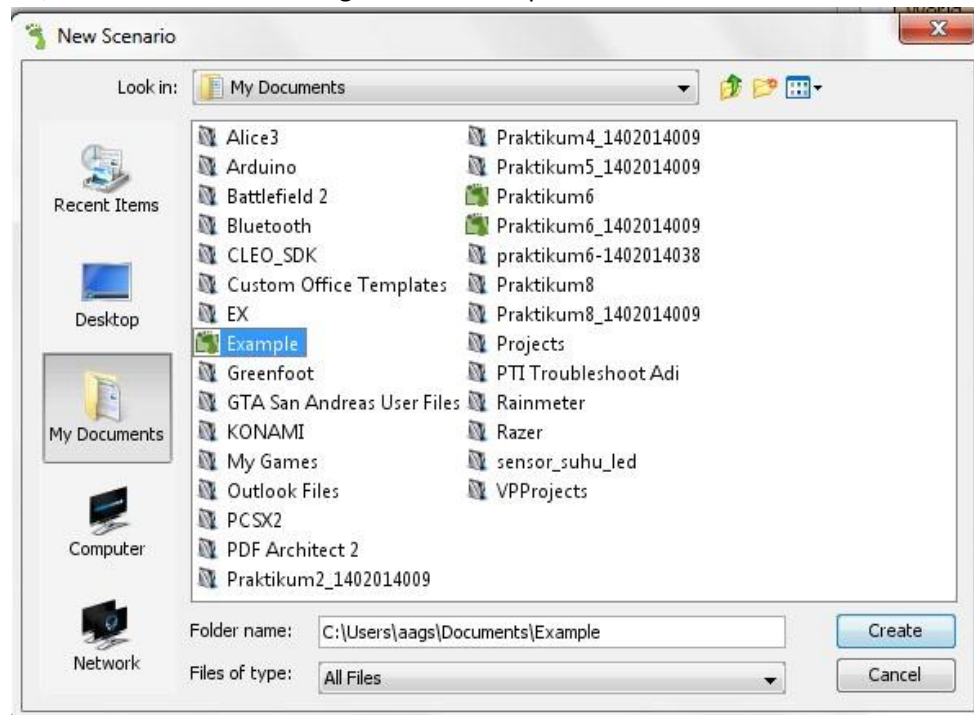
Bab 3 Membuat Game Sederhana

1. Buatlah suatu scenario dengan cara klik **new**



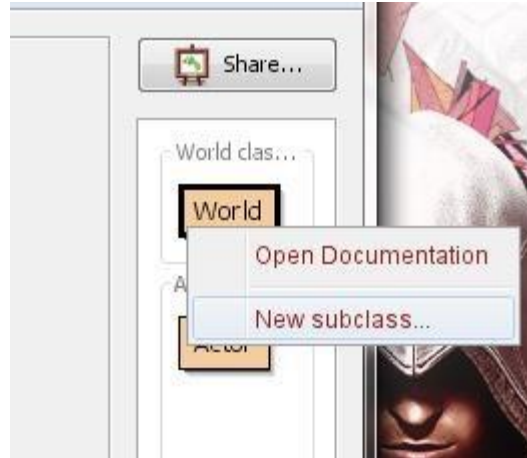
Gambar 34 membuat scenario

2. Setelah itu, buatlah folder baru dengan nama Example lalu klik **create**



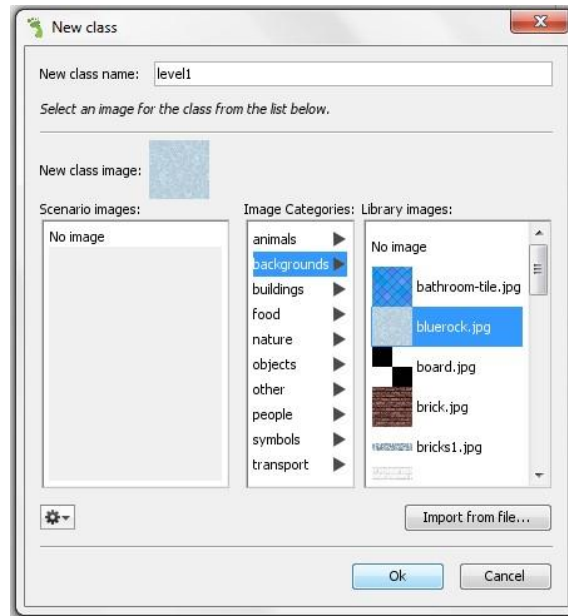
Gambar 35 membuat folder greenfoot

3. Setelah scenario dibuat, maka lihat sebelah kanan terdapat superclass World, kemudian klik kanan pada superclass World.



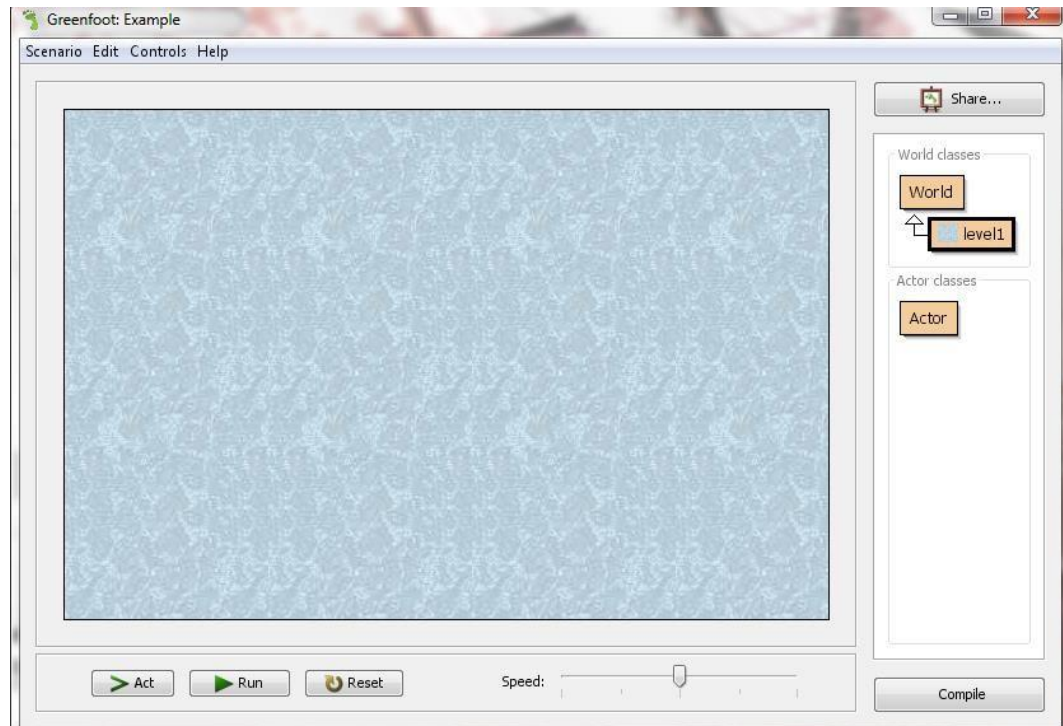
Gambar 36 membuat World baru

4. Akan muncul tampilan seperti gambar dibawah, klik background pada kolom Image Categories carilah gambar *bluerock.jpg* kemudian beri nama *level1*, lalu klik OK.



Gambar 37 memberi nama World

5. Setelah memberi nama pada world maka klik Compile di sebelah pojok kanan bawah, akan tampil gambar seperti dibawah ini.



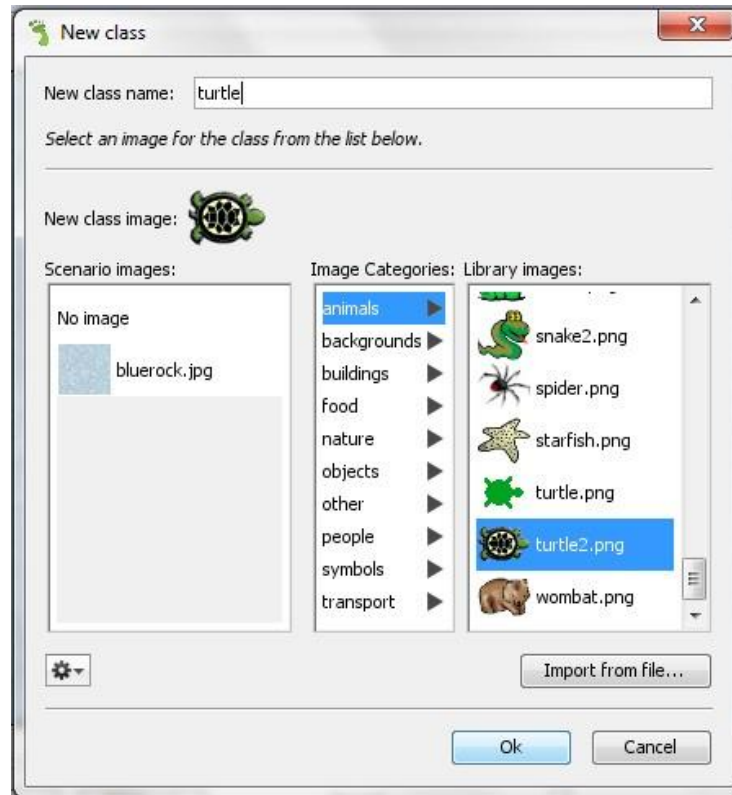
Gambar 38 tampilan world setelah di Compile

6. Tambahkan sebuah object, dengan cara klik kanan pada Actor lalu pilih new subclass



Gambar 39 menambahkan actor

7. klik animal pada kolom Image Categories carilah gambar *turtle2.png* kemudian beri nama *turtle* lalu klik OK.

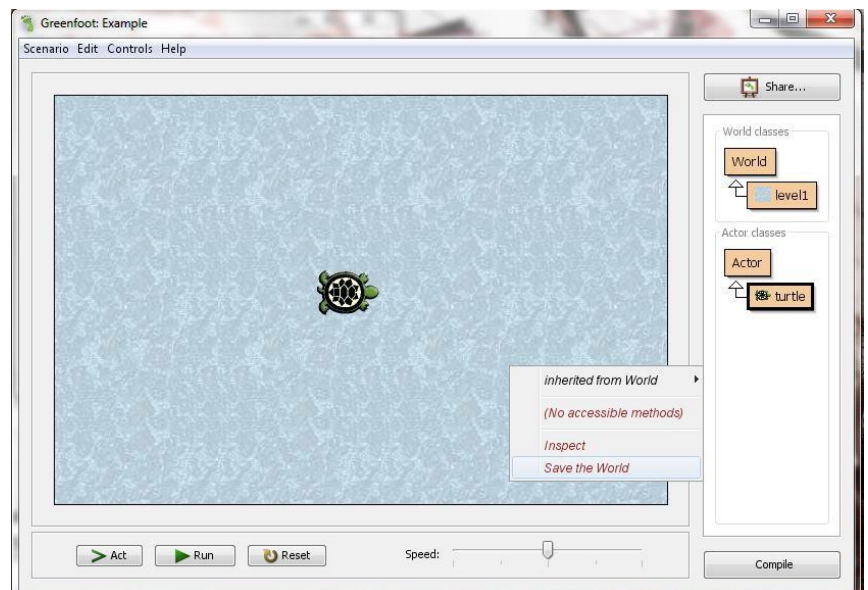


Gambar 40 memilih gambar object dan memberi nama

8. Setelah diberi nama, klik Compile. Tambahkan object kedalam world dengan mengklik kanan pada subclass *turtle* lalu klik *new turtle()* lalu drag object kedalam world. Setelah object dimasukkan kedalam world, klik kanan pada object lalu klik *Save the World*.



Gambar 42 menambahkan object kedalam world



Gambar 41 save the world agar object tidak hilang saat di compile



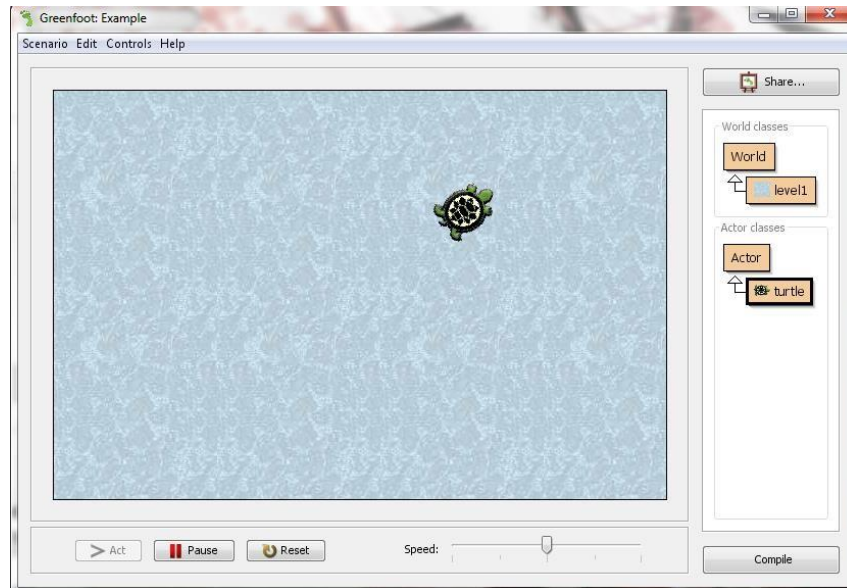
Gambar 43 membuka Source Code

```
public void act()
{
    // Add your action code here.
    move(1);
    if (Greenfoot.isKeyDown("left")){
        turn(-5);
    }
    if (Greenfoot.isKeyDown("right")){
        turn(5);
    }
}
```

9. Buatlah suatu method untuk menggerakkan *object turtle*, dengan cara klik kanan pada subclass *turtle* lalu klik *open editor*, kemudian tulislah perintah if yang berisi statement seperti pada gambar 11. Kemudian klik Compile maka akan muncul tulisan "*no syntax error*" dibagian paling bawah.

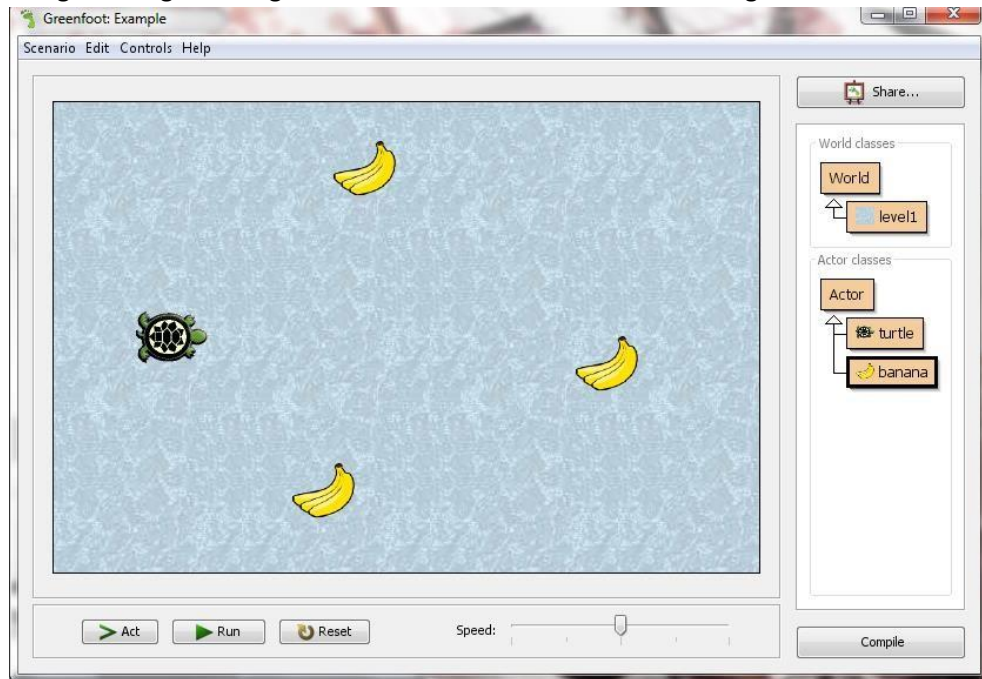
Gambar 44 method untuk menggerakkan object

10. Cobalah klik run kemudian tekan panah kanan atau kiri untuk menguji coba method yang telah dibuat.



Gambar 45 object bergerak

11. Tambahkan satu subclass lagi untuk makanan *turtle* carilah gambar pisang dan beri nama banana dengan mengikuti langkah 6-8. Tambahkan *banana* sesuai gambar dibawah ini.



Gambar 46 menambahkan object banana

12. Tambahkan statement dimana *turtle* memakan *banana* dengan membuka lagi Source Code *turtle*, kemudian tambahkan statement seperti pada gambar 14. Jangan lupa untuk mengcompile syntax.


```

public void act() |
{
    // Add your action code here.
    move(1);
    if (Greenfoot.isKeyDown("left")) {
        turn(-5);
    }
    if (Greenfoot.isKeyDown("right")) {
        turn(5);
    }
    Actor banana = getOneObjectAtOffset(0,0, banana.class);
    if (banana != null) {
        getWorld().removeObject(banana);
    }
}

```

Gambar 47 method untuk menghilangkan banana

13. Buatlah Score dengan mengikuti langkah ke 6, lalu beri nama *score* kemudian klik *No image* di kolom sebelah kiri. Setelah itu klik Ok.



Gambar 48 membuat score

14. Buka Source code score dengan klik kanan pada object score lalu klik *open editor*, kemudian tulislah method seperti gambar dibawah ini

```
public class score extends Actor
{
    /**
     * Act - do whatever the score wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    static int banana = 0;
    public void act()
    {
        // Add your action code here.
        GreenfootImage image = new GreenfootImage("Jumlah banana = "+banana, 36, Color.red, null);
        setImage(image);
    }
}
```

Gambar 49 method untuk membuat score

15. Buka kembali Source code *turtle* lalu tambahkan method "*score.banana++*"

```
public void act()
{
    // Add your action code here.
    move(1);
    if(Greenfoot.isKeyDown("left")){
        turn(-5);
    }
    if(Greenfoot.isKeyDown("right")){
        turn(5);
    }
    Actor banana = getOneObjectAtOffset(0,0, banana.class);
    if(banana != null){
        getWorld().removeObject(banana);
        score.banana++;
    }
}
```

Gambar 50 menambahkan method untuk menghitung score

16. Tambahkan object score kedalam World, klik kiri lalu pilih *new score()*, drag object score ke posisi pojok atas lalu klik kanan, kemudian klik "save the World".



Gambar 51 menambahkan score pada world

17. Tambahkan object baru untuk menjadi rintangan dalam game, ikuti langkah ke 6 lalu beri nama "bola " dengan memilih gambar "gold-ball.png" lalu klik ok. Setelah klik ok, jangan lupa untuk mengCompile.



Gambar 52 menambahkan object bola

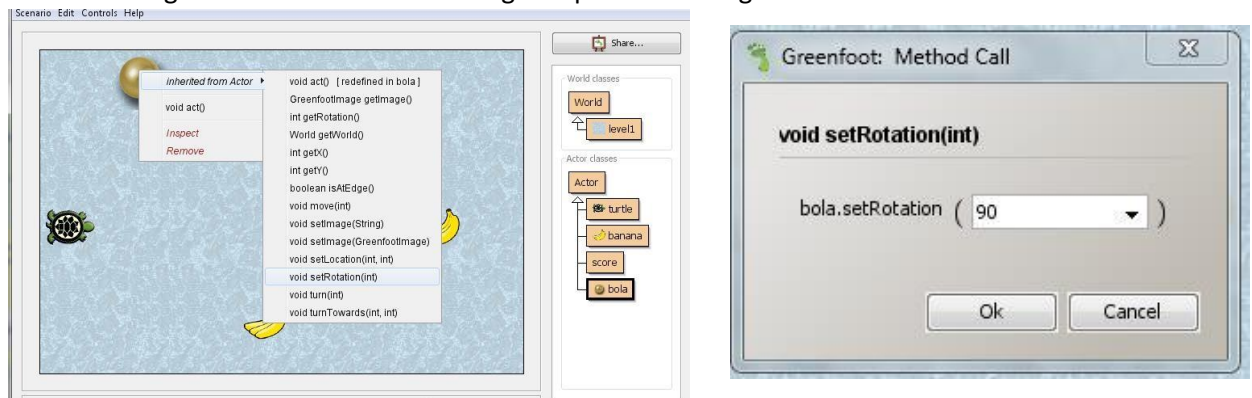
18. Buka Source code score dengan klik kanan pada object bola lalu klik *open editor*, kemudian tulislah method seperti gambar dibawah ini

```
import greenfoot.*;

/**
 * Write a description of class bola here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class bola extends Actor
{
    /**
     * Act - do whatever the bola wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(5);
        if(getY() <= 5 || getY() >= getWorld().getHeight() - 5)
        {
            turn(180);
        }
    }
}
```

Gambar 53 menggerakkan bola

19. Tambahkan object bola kedalam world dengan klik kiri *bola* lalu pilih *new bola()*, drag ke posisi atas. Agar bola bergerak kebawah maka klik kanan pada object bola lalu pilih *set Rotation*, masukkan angka 90 kemudian klik ok. Jangan lupa untuk mengklik *save the world*.



Gambar 54 method set rotation

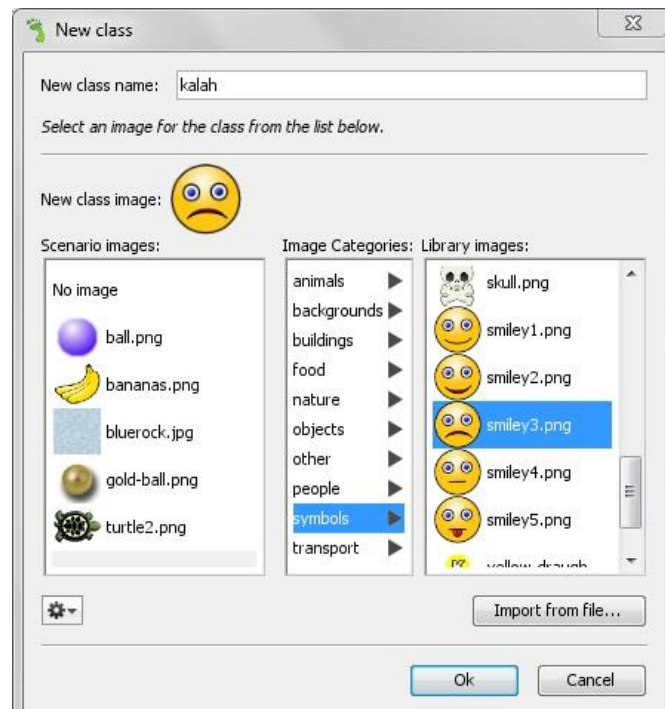
20. Buka sourcode bola, masukkan method untuk menghentikan game jika *bola* mengenai *turtle* dengan mengikuti gambar dibawah ini. Setelah itu compile.

```
import greenfoot.*;

/**
 * Write a description of class bola here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class bola extends Actor
{
    /**
     * Act - do whatever the bola wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(5);
        if(getY() <= 5 || getY() >= getWorld().getHeight() - 5)
            turn(180);
        Actor turtle = getOneObjectAtOffset(0, 0, turtle.class);
        if(turtle != null) {
            removeTouching(turtle.class);
            Greenfoot.stop();
        }
    }
}
```

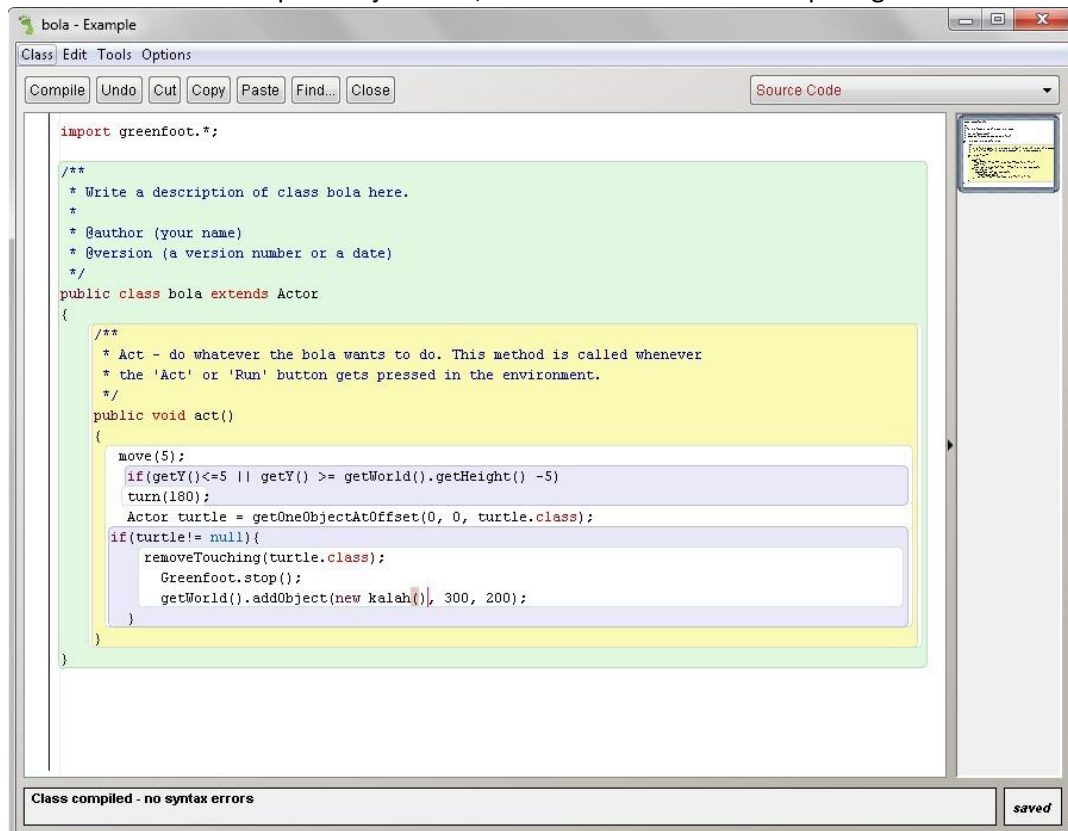
Gambar 55 method menghilangkan turtle dan memberhentikan game

21. Tambahkan object untuk menandakan jika game telah berakhir, ikuti langkah ke 6 lalu beri nama *kalah* dengan memilih gambar “smiley3.png” lalu klik ok. Setelah klik ok, jangan lupa untuk mengCompile.

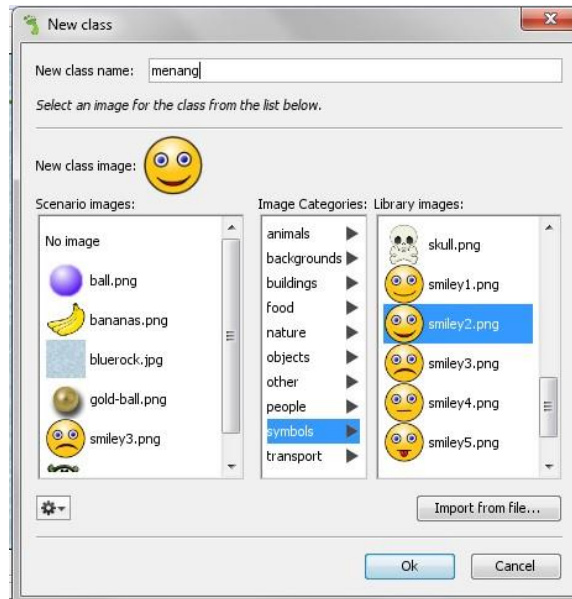


Gambar 56 menambahkan object kalah

22. Buka kembali source code pada object bola, lalu tambahkan method seperti gambar dibawah ini

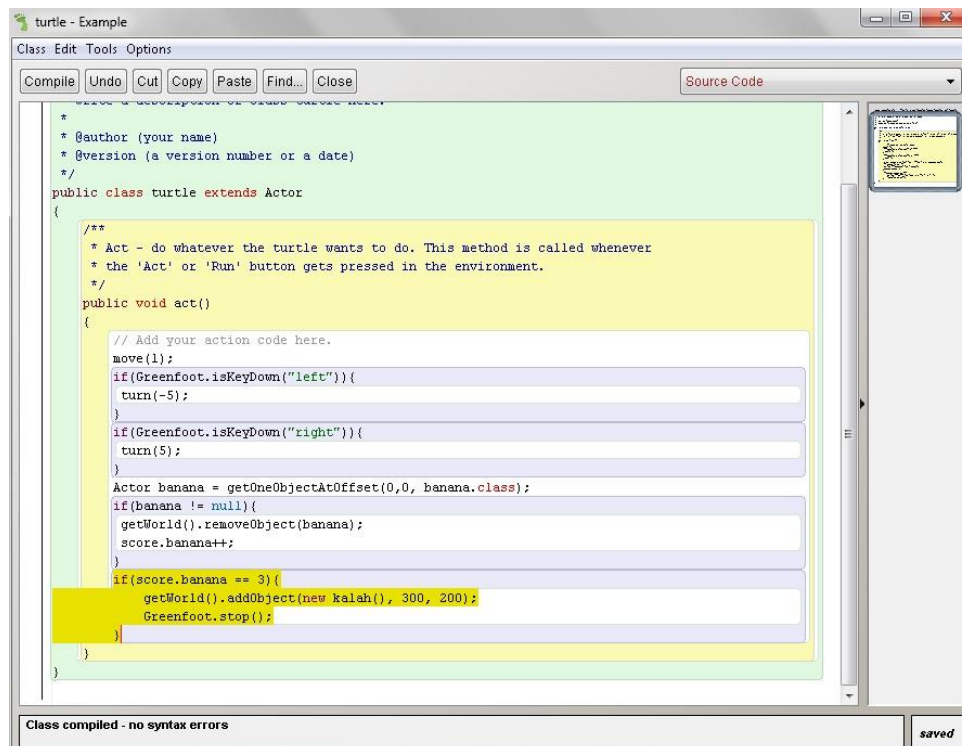


23. Tambahkan object untuk menandakan jika game telah menang, ikuti langkah ke 6 lalu beri nama *menang* dengan memilih gambar “smiley3.png” lalu klik ok. Setelah klik ok, jangan lupa untuk mengCompile.



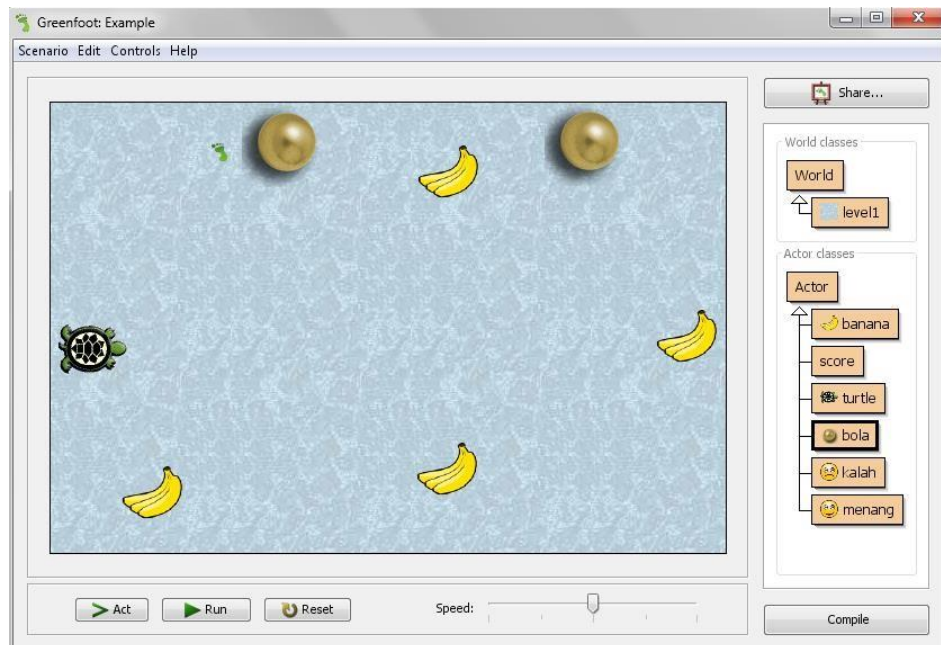
Gambar 58 menambahkan object menang

24. Buka kembali source code pada object *turtle*, lalu tambahkan method seperti gambar dibawah ini



Gambar 59 method untuk menang

25. Setelah semua langkah dilakukan kemudian klik run, gunakan keyboard panah kanan atau kiri untuk menggerakkan.



Gambar 60 tampilan akhir greenfoot