

# **MODUL PRAKTIKUM**

**Dasar-Dasar Pemrograman**



**Penyusun : Eko Verianto, S.Kom., M.Cs.**

# **DAFTAR ISI**

## **BAB 1**

PERTEMUAN 1 Pengantar Perkuliahan .	<b>Error! Bookmark not defined.</b>
PERTEMUAN 2 Pengenalan Alice .....	8

# PENGANTAR

# PERKULIAHAN

## Tujuan Pembelajaran:

1. Menjelaskan materi perkuliahan untuk satu semester
2. Menjelaskan tujuan pembelajaran akhir
3. Menjelaskan mengenai Oracle Academy sebagai kurikulum penunjang kebutuhan pembelajaran
4. Menjelaskan peta pembelajaran
5. Menjelaskan program aplikasi yang digunakan selama perkuliahan
6. Pengenalan Alice 3

## Materi Perkuliahan

1. Pengantar Perkuliahan
2. Memulai dengan Alice
3. Memulai dengan Greenfoot
4. Dasar Pemrograman Java
5. Struktur dalam Pemrograman Java
6. Array dan Exception
7. Class pada Java

## Tujuan Pembelajaran Akhir

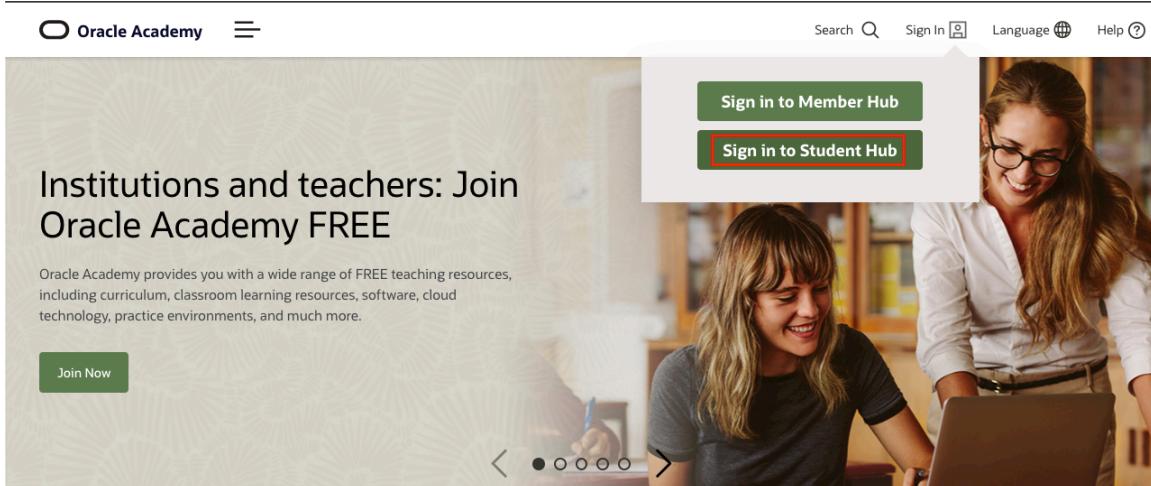
Mahasiswa akan mempelajari mengenai elemen dasar pada program komputer dan juga bahasa pemrograman java untuk memulai menulis kode program pada aplikasi yang akan dikembangkan. Di akhir perkuliahan diharapkan mahasiswa dapat:

1. Membuat animasi dan permainan sederhana
2. Mendemonstrasikan pengetahuan yang didapatkan terkait teknologi java dan juga bahasa pemrograman java
3. Menggunakan bahasa pemrograman java untuk membuat aplikasi
4. Mengintegrasikan struktur percabangan, perulangan dan teknik lanjutan lainnya untuk membuat aplikasi

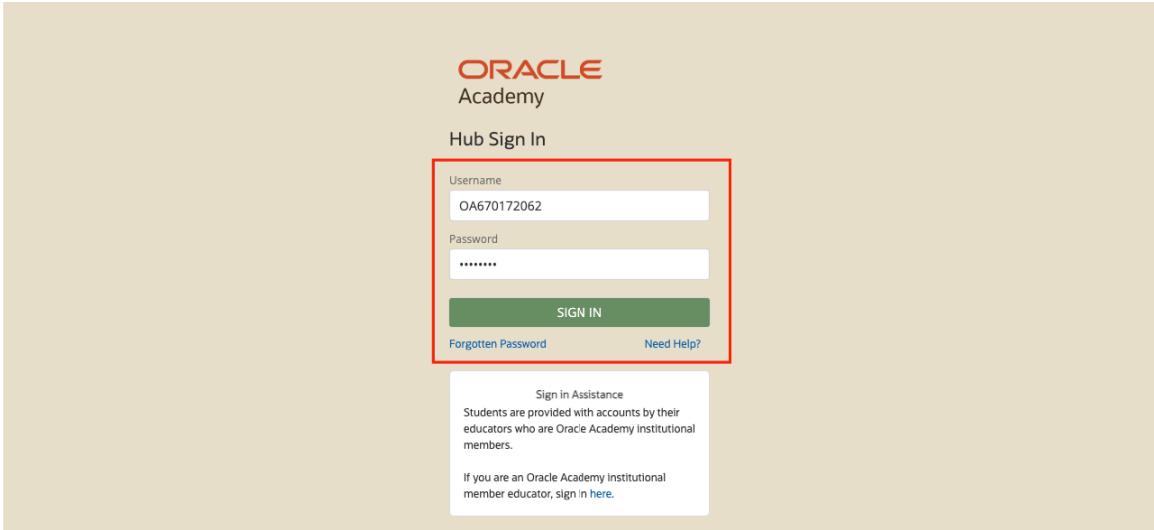
## Kurikulum Oracle Academy

Selama satu semester ini, kita akan menggunakan kurikulum dari oracle academy untuk menunjang proses pembelajaran kita. Anda dapat mengikuti langkah-langkah berikut ini untuk membuka akun oracle academy anda.

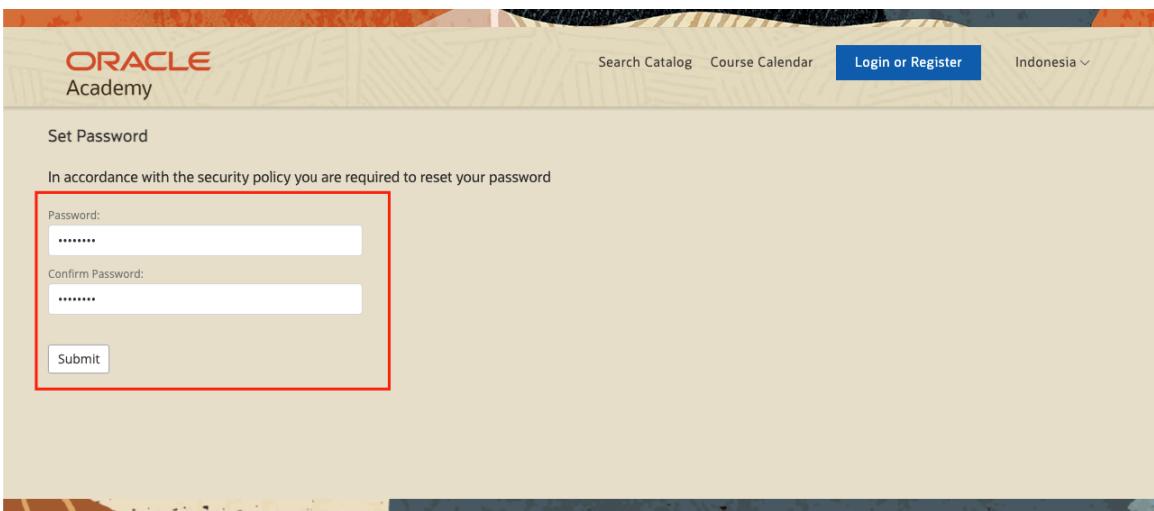
1. Silahkan buka tautan berikut ini <https://academy.oracle.com/>
2. Pilih menu sign in kemudian pilih sign in to student hub



3. Masukan username dan password sesuai dengan akun anda (cek pada daftar)



4. Setelah berhasil sign in, anda diminta untuk mengubah password anda, silahkan lakukan perubahan pada password (gunakan password yang biasa digunakan)



5. Setelah selesai mengubah password, anda akan diarahkan ke halaman utama dashboard. Pada halaman ini terdapat satu channel anda dengan nama "praktikum dasar-dasar pemrograman". Klik pada channel dengan nama "praktikum dasar-dasar pemrograman" tersebut.

The screenshot shows the Oracle Academy homepage. At the top right, there is a user profile for 'Eko Verianto 72220000' labeled 'Learner'. The main navigation bar includes 'Home', 'Career Center', 'My Learning', and 'My Reports'. A banner on the left says 'Explore careers!' and 'Career Center'. On the right, there is a 'More Help' sidebar with links for 'Redeem Certification Exam Vouchers Demonstration' and 'Member Hub Learner Guide'. Below the banner, under 'My Channels', there is a list with one item: 'Praktikum Dasar-Dasar Pemrograman' with a 'Details' button. This list is highlighted with a red border.

6. Setelah memilih channel, anda akan diarahkan ke halaman learning path. Pada bagian ini anda memiliki satu learning path dengan nama "JF Java Fundamentals Learner – English". Silahkan klik pada learning path dengan nama "JF Java Fundamentals Learner – English" tersebut

The screenshot shows the Oracle Academy homepage again. The 'Learning Paths' section is highlighted with a red border. It displays a card for 'Java Fundamentals Learner' with a small icon, the text 'Learning Path - Full Course', and 'JF Java Fundamentals Learner - English'. There is also a 'Details' button at the bottom right of the card.

7. Selanjutnya anda akan masuk ke halaman course yang berisi materi pembelajaran, soal kuis serta soal TTS dan TAS. Pada halaman ini anda dapat melihat dan mengunduh materi.

The screenshot shows the Oracle Academy homepage. At the top right, there is a user profile for "Eko Verianto 72220000" with the status "Learner". The main navigation bar includes links for "Home", "Career Center", "My Learning", and "My Reports". Below the navigation, there is a vertical list of course sections: "Section 0 - Course Resources", "Section 1 - Introduction", "Section 2 - Alice 3", and "Section 3 - Greenfoot".

- a. Untuk melihat materi, anda dapat memilih course terlebih dahulu, kemudian klik play

The image contains three screenshots of the Oracle Academy interface:

- Screenshot 1:** Shows the "Section 1 - Introduction" section selected. A specific video thumbnail titled "JF 1-1: Introduction" is highlighted with a red box.
- Screenshot 2:** Shows the "JF 1-1: Introduction" video details page. The "Play" button is highlighted with a red box.
- Screenshot 3:** Shows the video player interface for "JF 1-1: Introduction". The video title "JF 1-1: Introduction" is visible, along with the "Online" status and "Credits (HH:MM):00:48".

b. Untuk mengunduh materi, anda dapat memilih course terlebih dahulu, kemudian klik nama dokumen pada bagian reference material

The screenshot shows the Oracle Academy interface. At the top, there's a navigation bar with links for Home, Career Center, My Learning, and My Reports. On the right, a user profile for 'Eko Verianto 72220000 Learner' is visible. Below the navigation, a course card for 'JF 1-1: Introduction' is displayed. The card includes a thumbnail of a person riding a motorcycle, the title 'Java Fundamentals Learner', status 'Online', credits '(HH:MM):00:48', and a 'Play' button. To the right of the card, there's a 'Details' section and another card for 'JF 1-1: Introduction' with a red box around the PDF download link 'JF 1-1: Introduction (SG PDF) (466.8 KB)'.

c. Untuk mengerjakan kuis, TTS dan TAS, anda dapat memilihnya *quiz*, *midterm exam* atau *final exam* terlebih dahulu, kemudian klik play

The screenshot shows the Oracle Academy interface with a list of quizzes and an exam detail page. At the top, there's a navigation bar with links for Home, Career Center, My Learning, and My Reports. Below the navigation, a list of items is shown, each with a thumbnail of a person riding a motorcycle, the title 'Java Fundamentals Learner', status 'Online', credits '(HH:MM):00:30', and a 'Play' button. The first item in the list is highlighted with a red box around its title 'JF Section 2 Quiz 1 - L1-L7'. Below the list, a detailed view of the quiz is shown with a red box around the 'Play' button. The detailed view includes the title 'JF Section 2 Quiz 1 - L1-L7', status 'Exam', credits '(HH:MM):00:30', and a 'Details' section. To the right of the detailed view, there's another card for 'JF Section 2 Quiz 1 - L1-L7' with a red box around the PDF download link 'JF Section 2 Quiz 1 - L1-L7 (SG PDF) (466.8 KB)'.

## **Peta Pembelajaran**

1. Pengantar Perkuliahan, Pengenalan Alice 3, Menambahkan & Memposisikan Objek, Procedure & Argument
2. Rotation & Randomization, Deklarasi Procedure, Control Statement, Function
3. Kuis, Struktur Kontrol IF & WHILE, Ekspresi, Variabel, Kontrol Keyboard
4. Membuat animasi sederhana, Variabel & Tipe Data, Class & Method, Quiz
5. Pengenalan GreenFoot, Method, Variabel & Parameter, Source Code & Documentation
6. Mengembangkan dan Menguji Aplikasi, Randomization & Memahami Notasi Dot, Konstruktor, Quiz
7. Mendefinisikan Metode, Suara dan Kontrol Keyboard, World Animation, Abstraction, Perulangan, Variabel dan Array, Quiz
8. Midterm Exam
9. Pengenalan Eclipse, Driver Class & Object Class, Tipe Data & Operator, String, Quiz
10. Scanner & Conditional Statement, Control Statement, Quiz
11. Array, Handling Error, Quiz
12. Classes, Object, & Method, Static Modifier
13. Nested Classes, Inheritance
14. Polymorphism, Quiz

## **Program Aplikasi**



# PENGENALAN ALICE

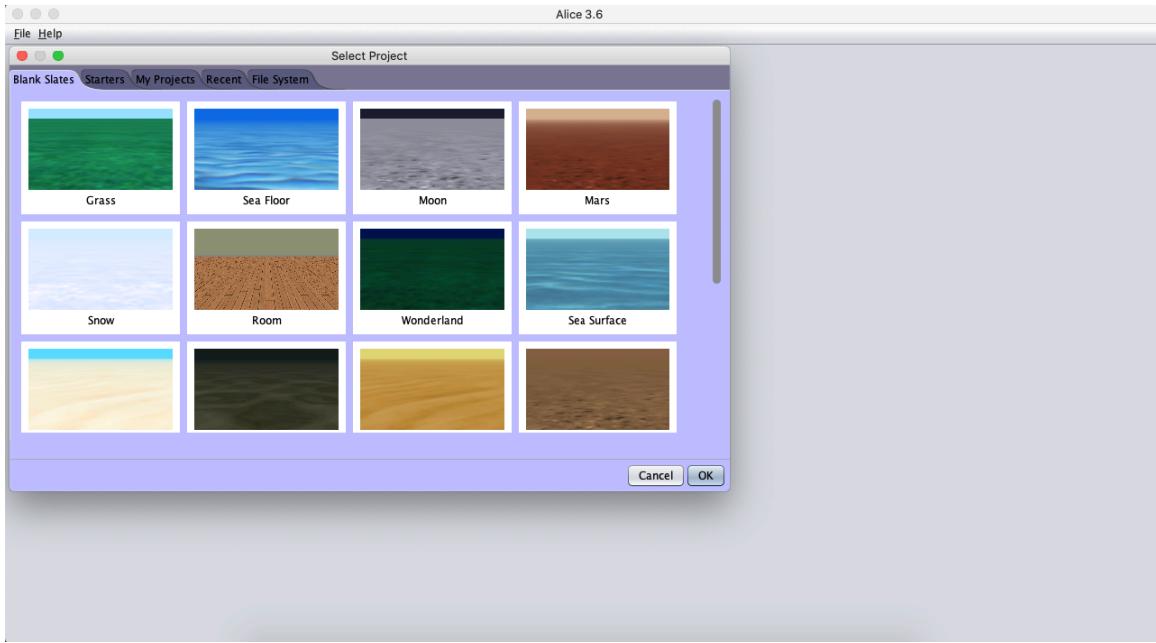
## Tujuan Pembelajaran:

1. Menjelaskan materi perkuliahan untuk satu semester
2. Menjelaskan tujuan pembelajaran akhir
3. Menjelaskan mengenai Oracle Academy sebagai kurikulum penunjang kebutuhan pembelajaran
4. Menjelaskan peta pembelajaran
5. Menjelaskan program aplikasi yang digunakan selama perkuliahan

## Pengenalan Alice 3

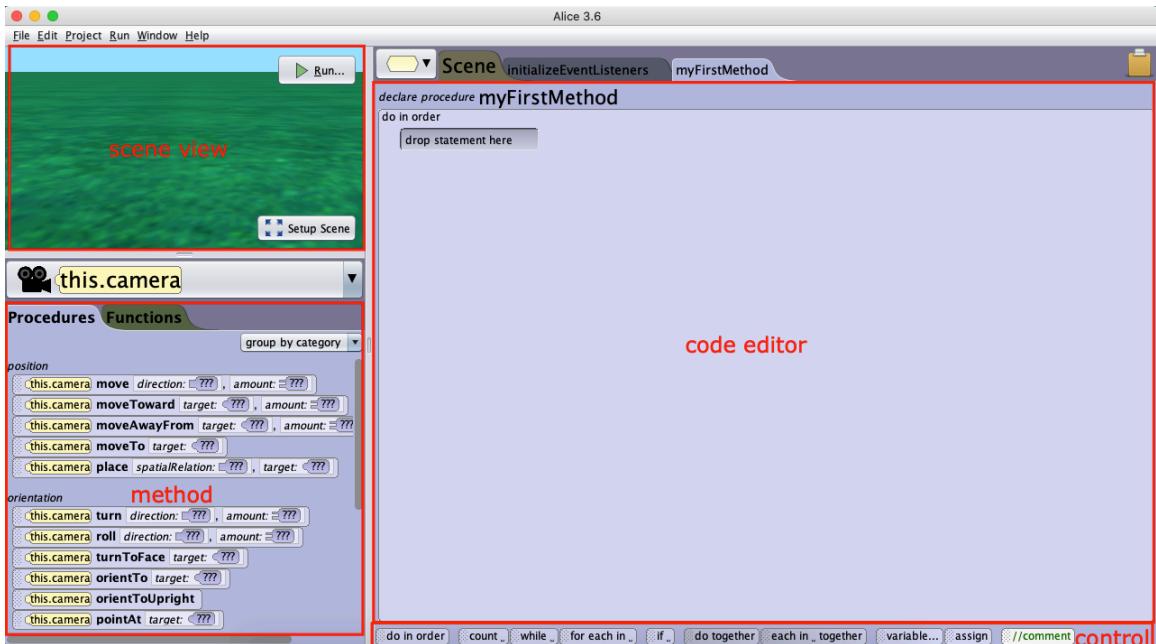
Alice 3 (tiga) merupakan pengembangan dari versi sebelumnya, ini merupakan pemrograman 3D dan freeware dari bahasa pemrograman berbasis objek yang digunakan untuk membuat prototipe environment dalam membangun dunia virtual yang sederhana. Alice juga digunakan untuk media pembelajaran pemrograman berorientasi objek. Lakukan percobaan berikut ini:

1. Silahkan buka aplikasi Alice 3, jika berhasil maka tampilan awalnya seperti berikut ini:

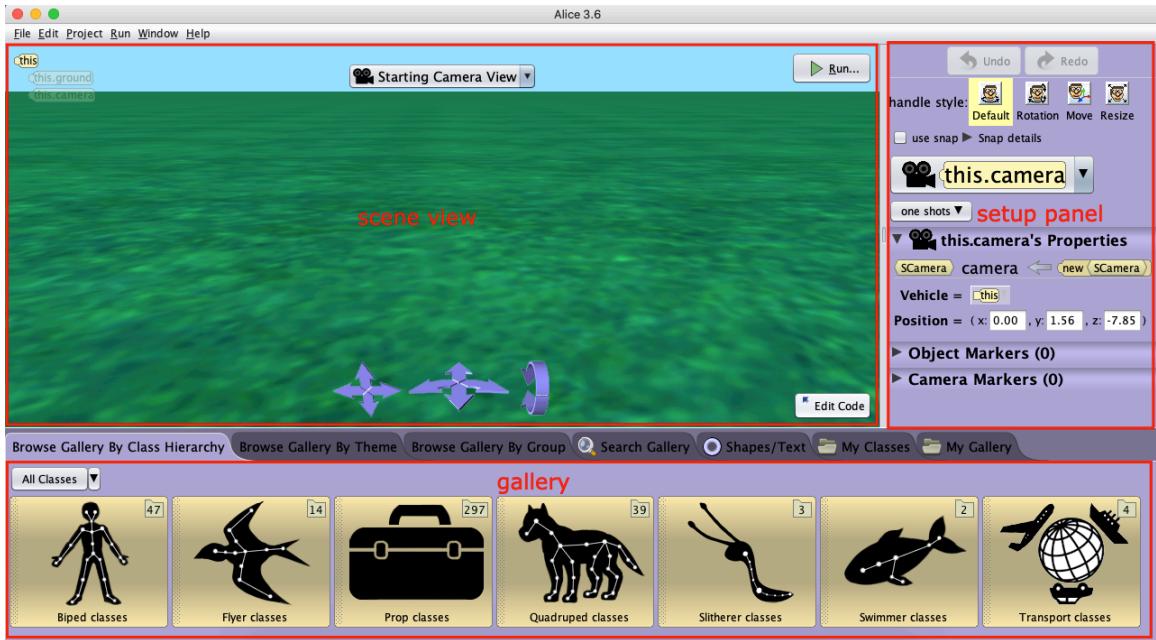


Pada bagian ini terdapat beberapa latar belakang yang dapat anda gunakan, selanjutnya pilih salah satu latar belakang yang anda inginkan kemudian klik ok.

## 2. Tampilan Editor

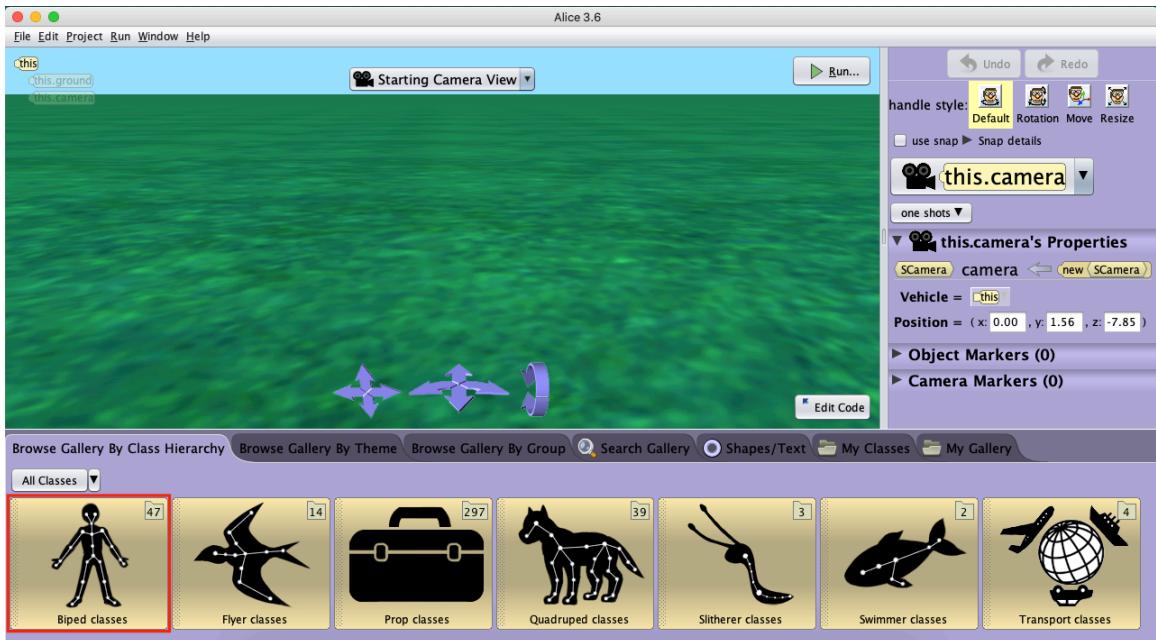


## 3. Tampilan Setup Scene



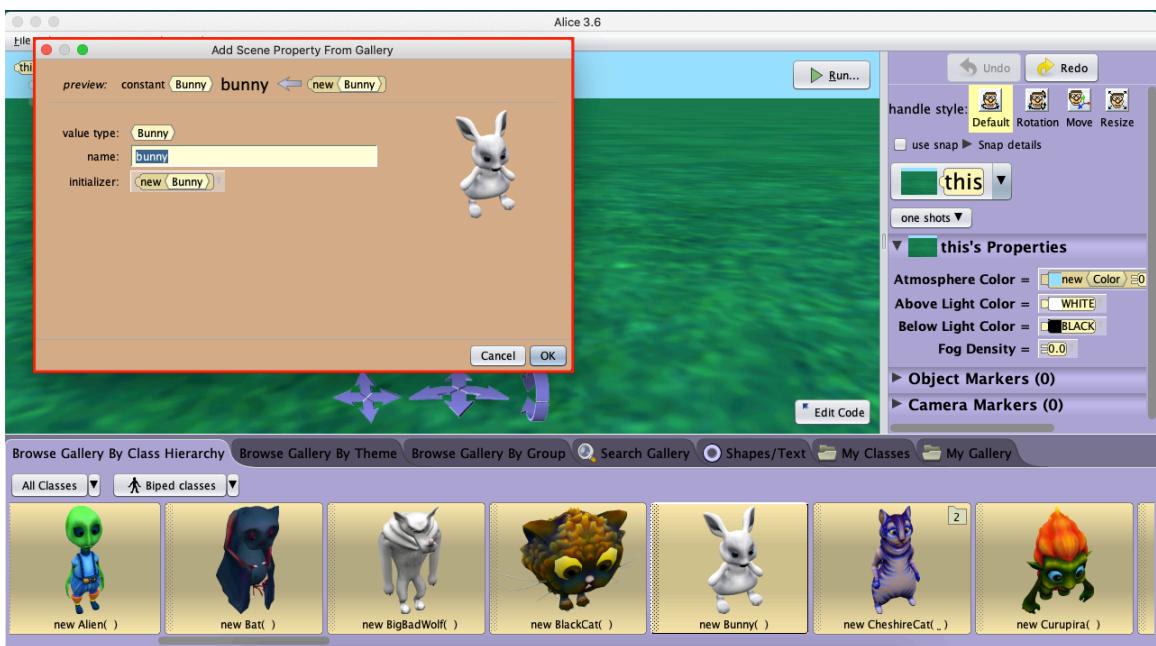
#### 4. Menambahkan Objek

Pilih salah satu kategori class pada galeri yang disediakan, misalkan saja kita memilih biped, kemudian klik class yang diinginkan misalkan saja bunny, atau drag and drop class bunny ke bagian scene view





5. Jika berhasil maka akan ada jendela pop up untuk mengatur properti pada class



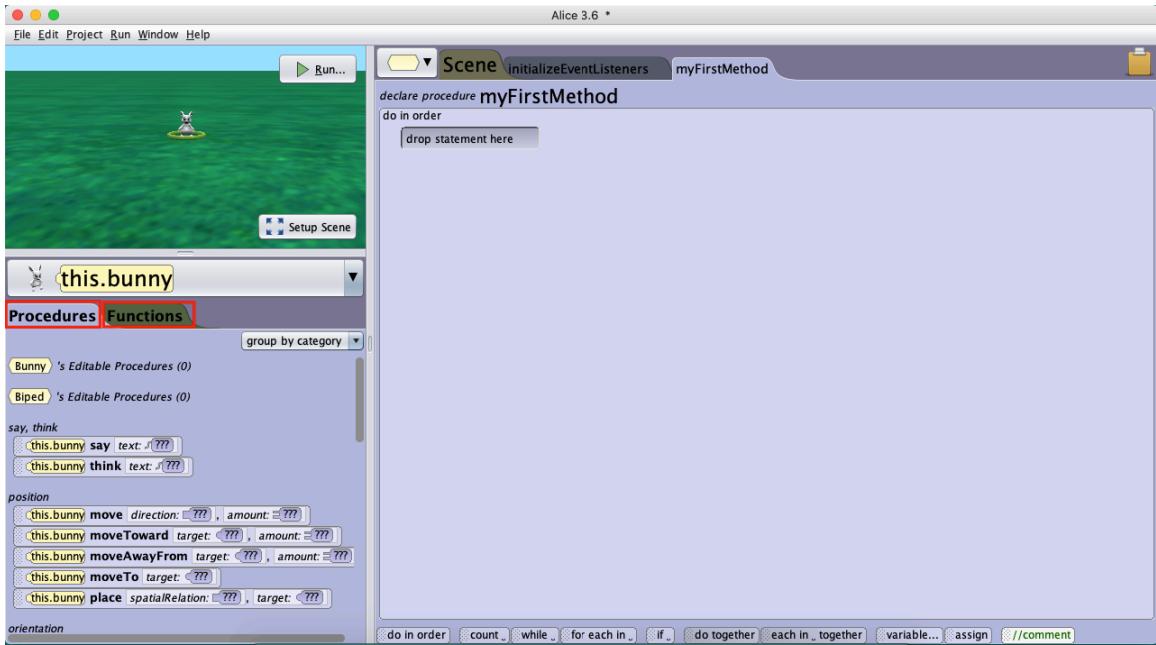
6. Jika sudah selesai mengelola properti class maka akan tercipta sebuah objek baru pada scene view



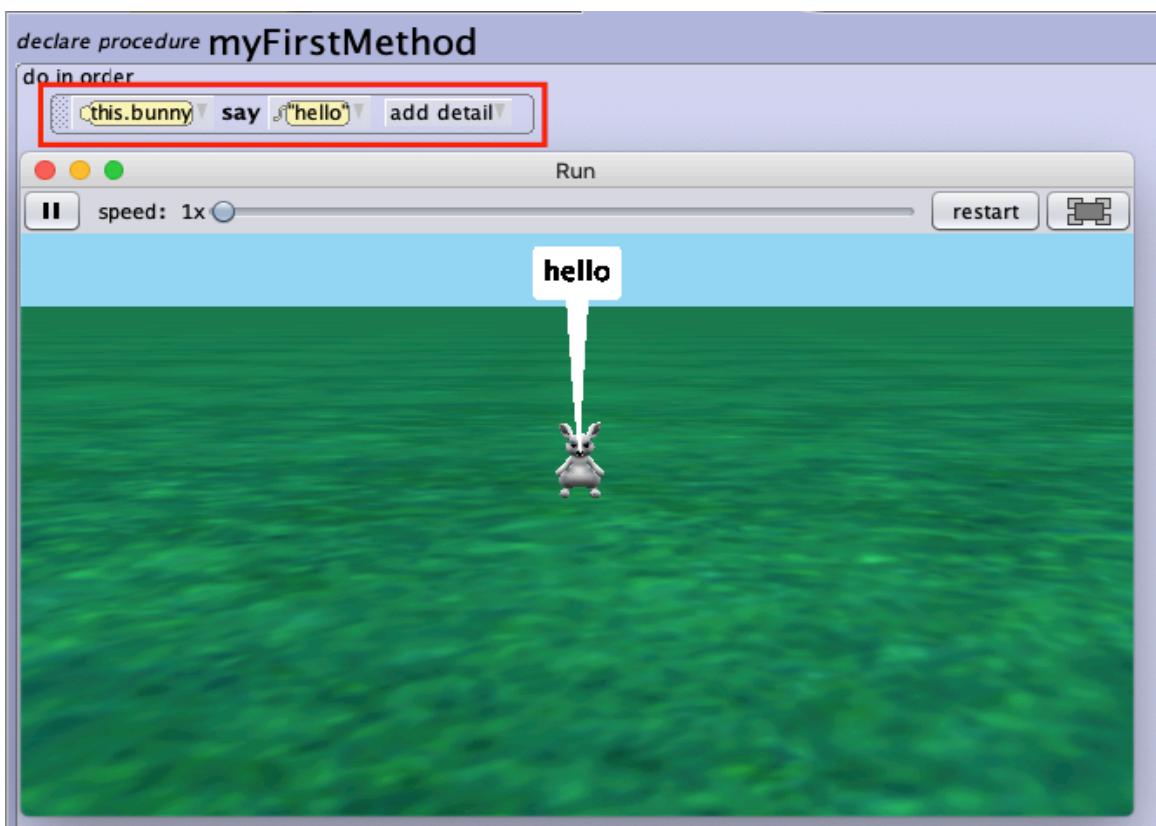
7. Kembali ke code editor dengan klik edit code untuk menambahkan beberapa instruksi



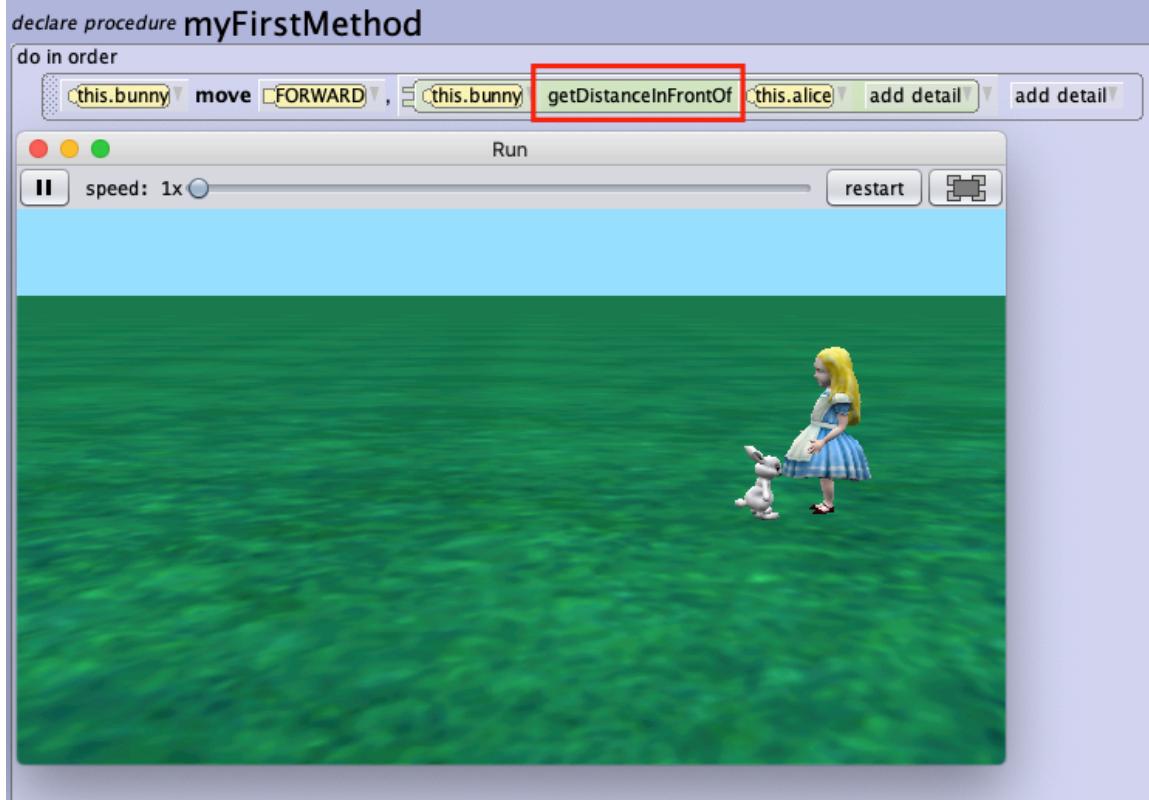
8. Terdapat dua metode yang dapat digunakan yang tersedia pada panel metode, diantaranya adalah procedures dan functions. Procedures merupakan metode yang tidak dapat mengembalikan nilai, sedangkan functions adalah metode yang mengembalikan nilai.



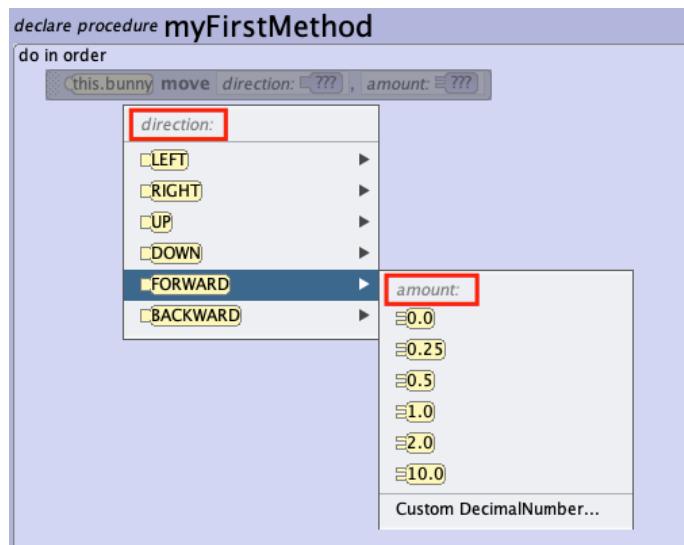
9. Penggunaan procedure akan langsung dieksekusi sesuai dengan instruksi dari metode tersebut, sebagai contoh jika procedure say ditambahkan pada code editor, ketika program dijalankan maka instruksi dari procedure tersebut langsung dieksekusi



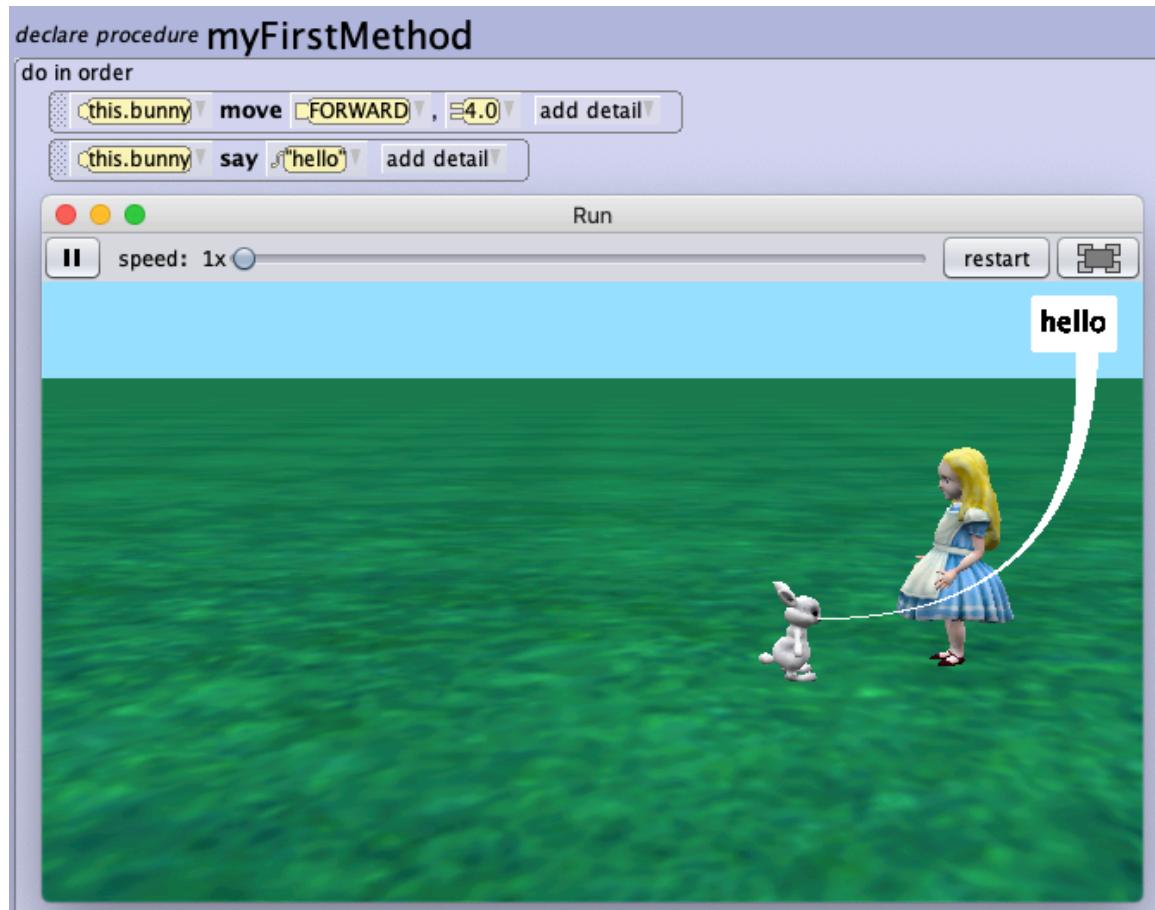
10. Penggunaan function tidak dapat langsung ditambahkan pada code editor, anda harus menyisipkan function pada sebuah procedure. Hal ini dikarenakan sifat dari function yang mengembalikan nilai



11. Pada alice terdapat argumen, dalam pemrograman argumen merupakan nilai yang diberikan. Pada alice terdapat beberapa tipe argumen diantaranya adalah direction, amount, duration, text.



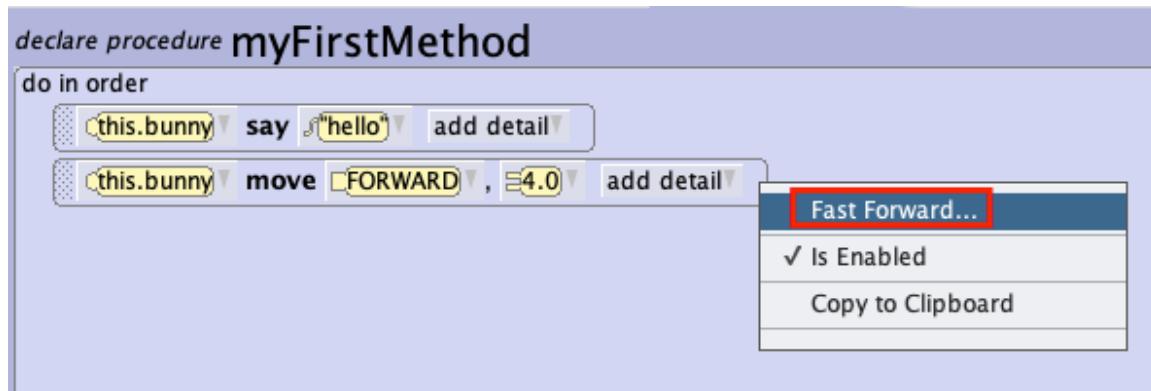
12. Menambahkan lebih dari satu instruksi secara terarah dengan menggunakan kontrol do in order



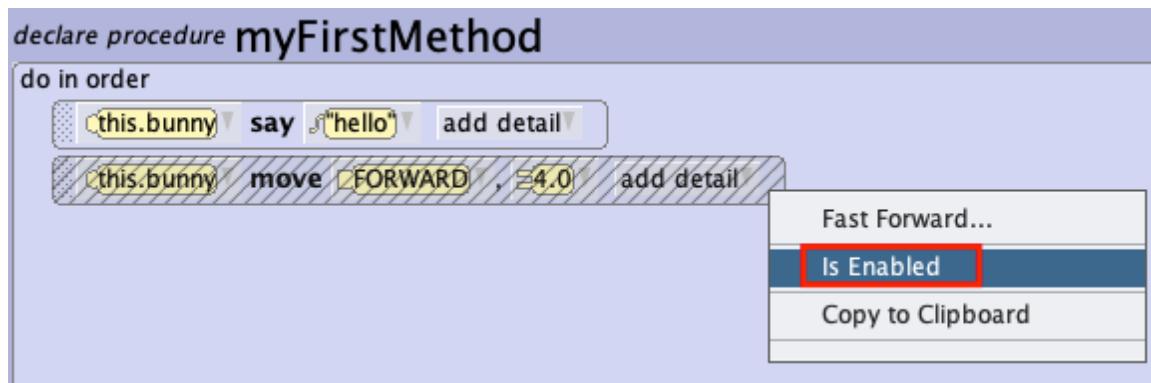
13. Mengubah urutan procedure pada code editor



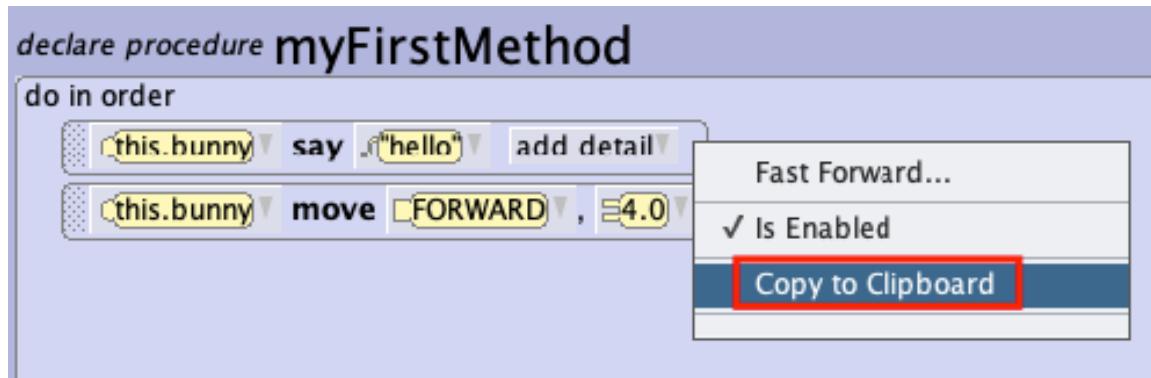
14. Menjalankan animasi dari instruksi atau procedure tertentu



15. Enable dan disable instruksi atau procedure

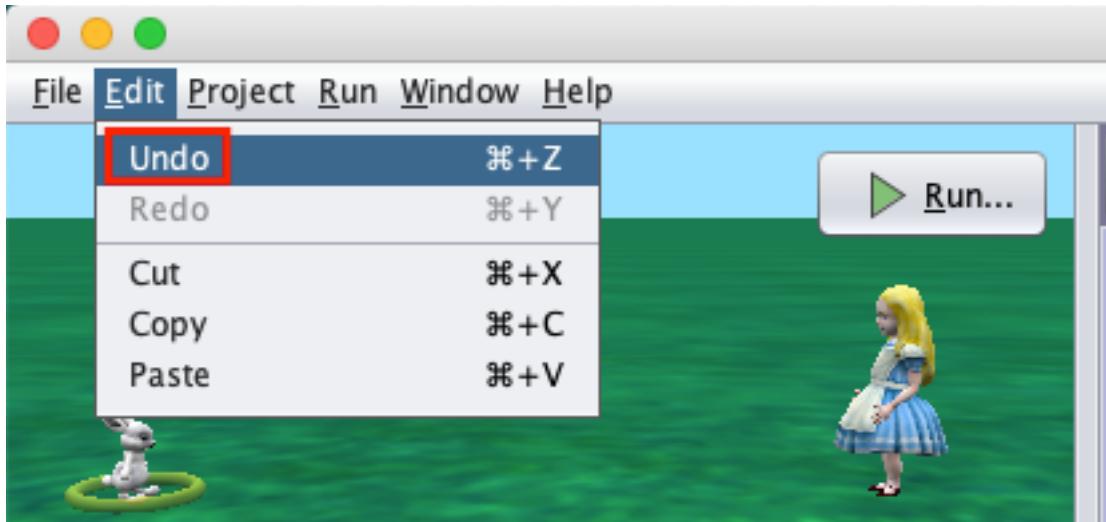


16. Menyalin instruksi pada clipboard

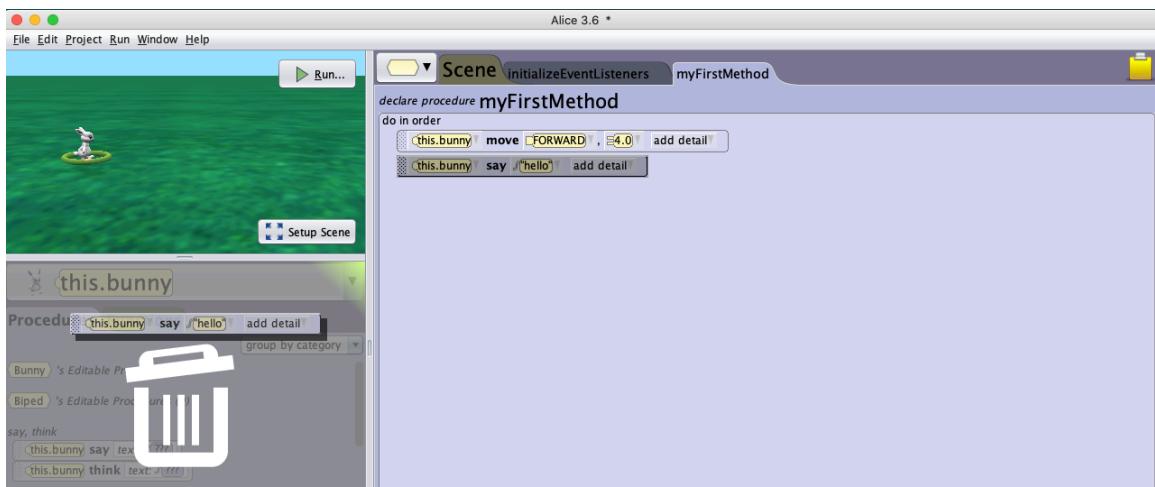


17. Membatalkan perintah

Instruksi ataupun aktivitas sebelumnya dapat dibatalkan dengan undo



### 18. Menghapus instruksi



### 19. Menambahkan Komentar



### 20. Melakukan testing dan debugging

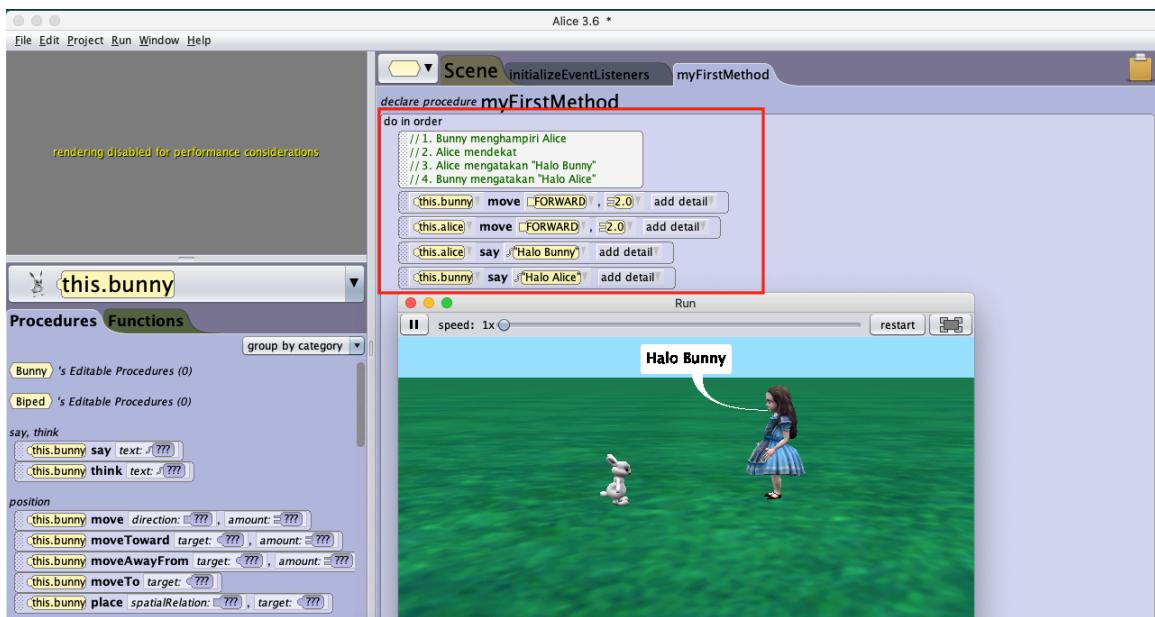
Beberapa teknik yang dapat digunakan diantaranya sesuaikan argumen, sesuaikan ekspresi matematika, perbaiki metode atau instruksi, dan mengatasi masalah lainnya

21. Simpan proyek dengan klik file kemudian klik save, tentukan nama proyek dan direktori tempat anda menyimpan proyek ini, selanjutnya klik save

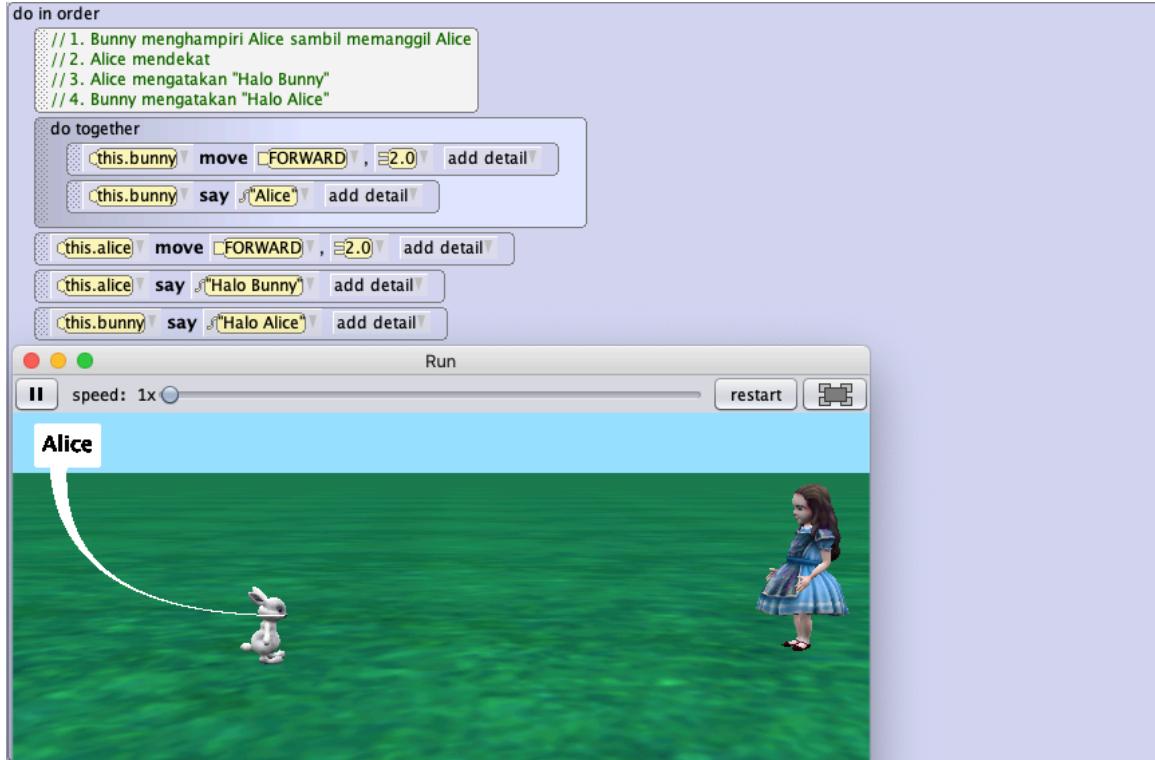


22. Proses pengembangan program dapat dilakukan dengan menggunakan pendekatan top-down, artinya programmer bekerja dengan tujuan. Langkah untuk menggunakan pendekatan top-down adalah dengan membuat dokumen langkah demi langkah dalam bentuk teks (storyboard: Modul JF 2.5 SG). Berikut ini merupakan contoh pembuatan storyboard pada alice 3 dalam bentuk teks.

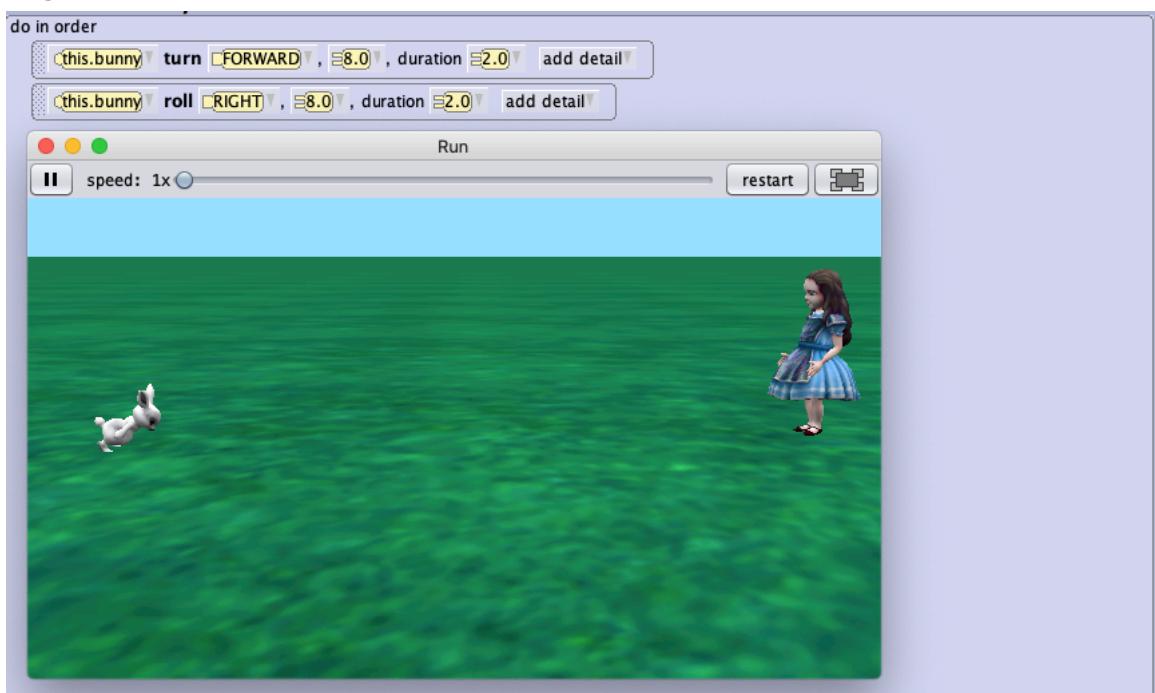
- 1) Bunny menghampiri alice
- 2) Alice mendekat
- 3) Alice mengatakan "Halo Bunny"
- 4) Bunny mengatakan "Halo Alice"



Menggunakan control do together pada program alice



23. Perbedaan procedure turn dan roll, rotasi objek pada procedure turn ke arah kiri, kanan, depan dan belakang dari titik pusat objek, sedangkan roll objek bergulir ke arah kiri dan kanan dari titik pusat objek.



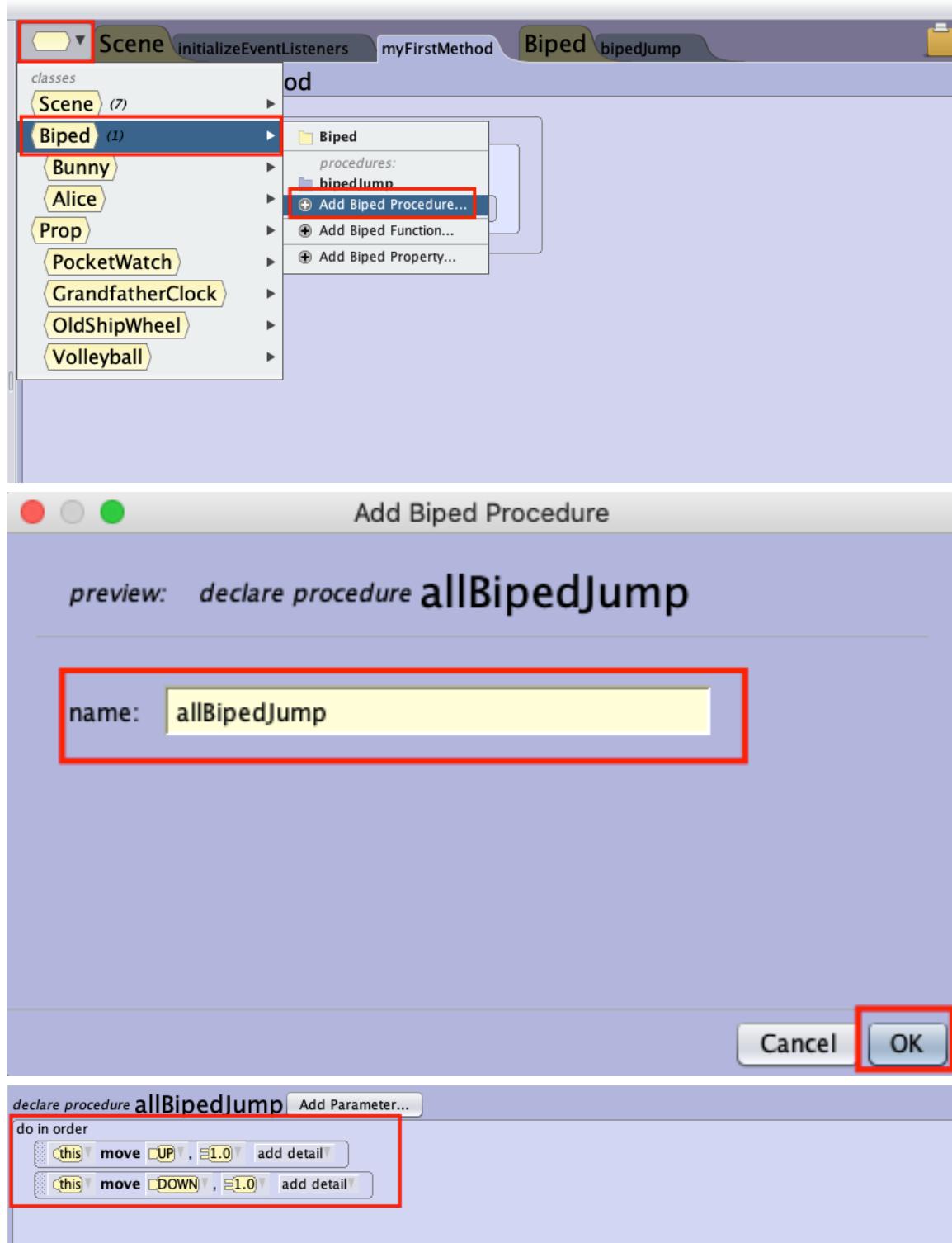
## 24. Mengendalikan sub-part pada objek

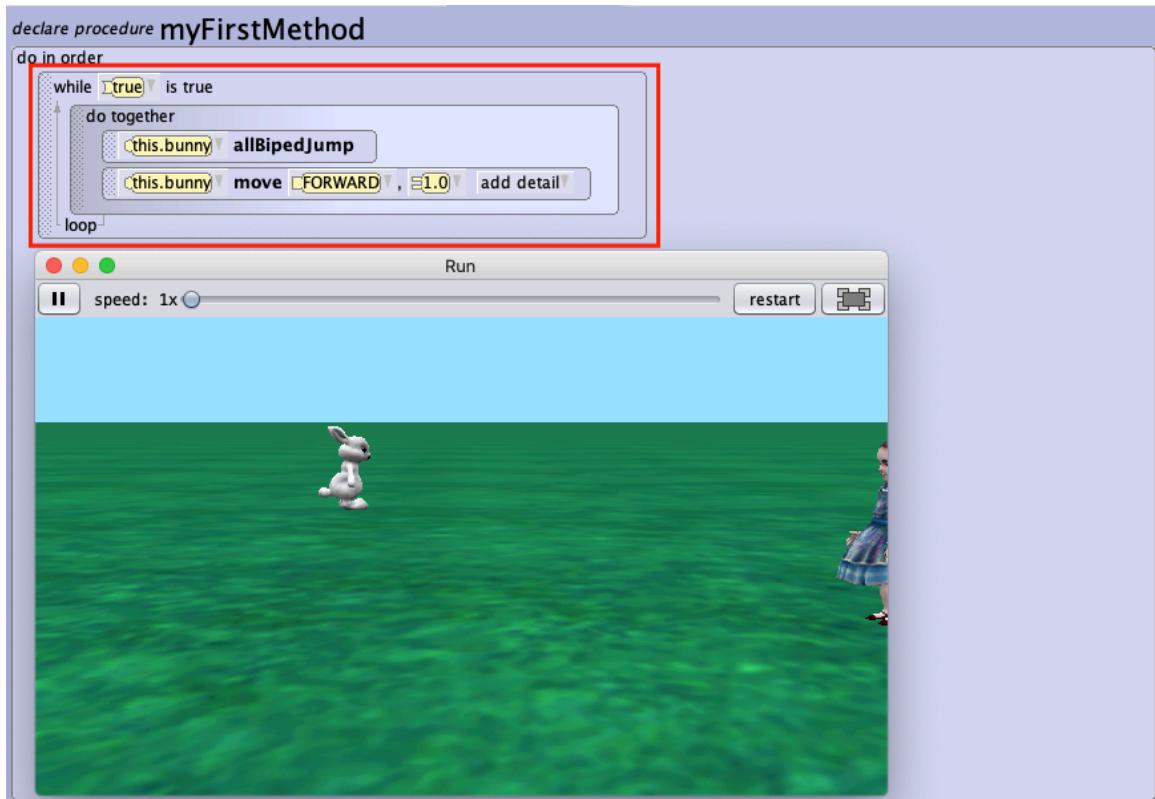


## 25. Menambahkan procedure pada control statement & nested do together



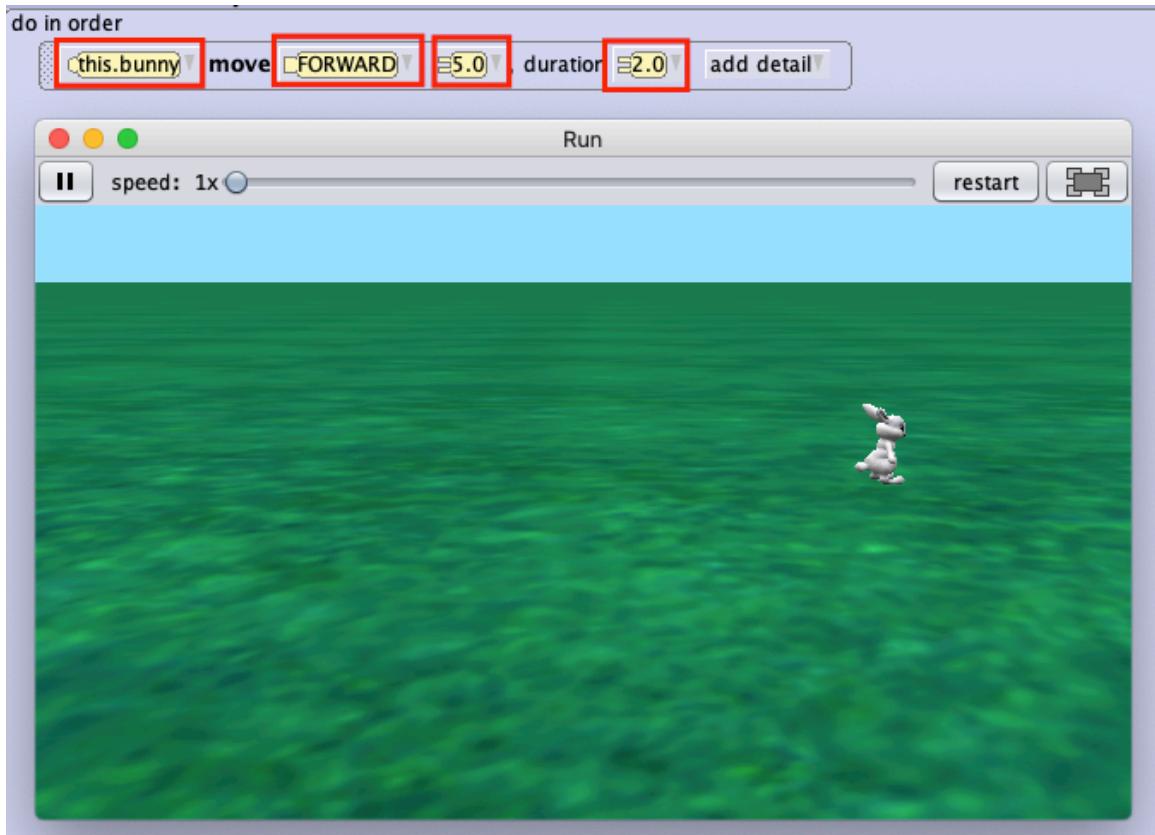
26. Deklarasi Procedure (membuat metode procedure). Procedure dibuat ketika terjadi procedure abstraction (instruksi berulang/bagian dari instruksi yang dilakukan secara berulang)



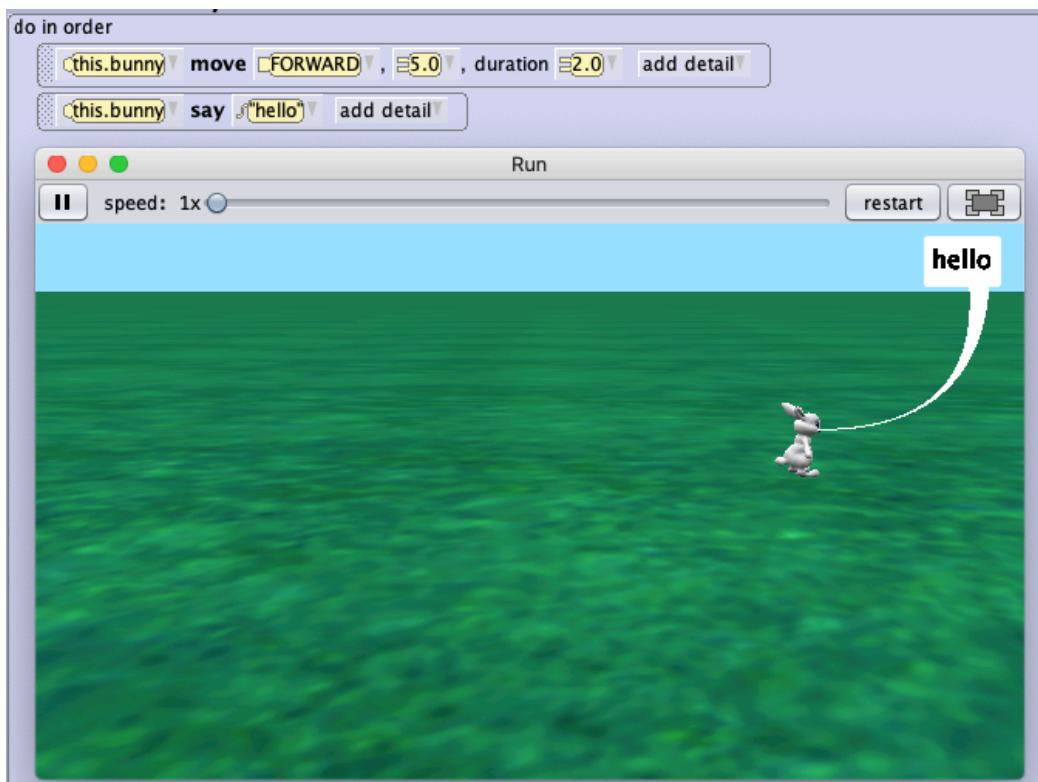


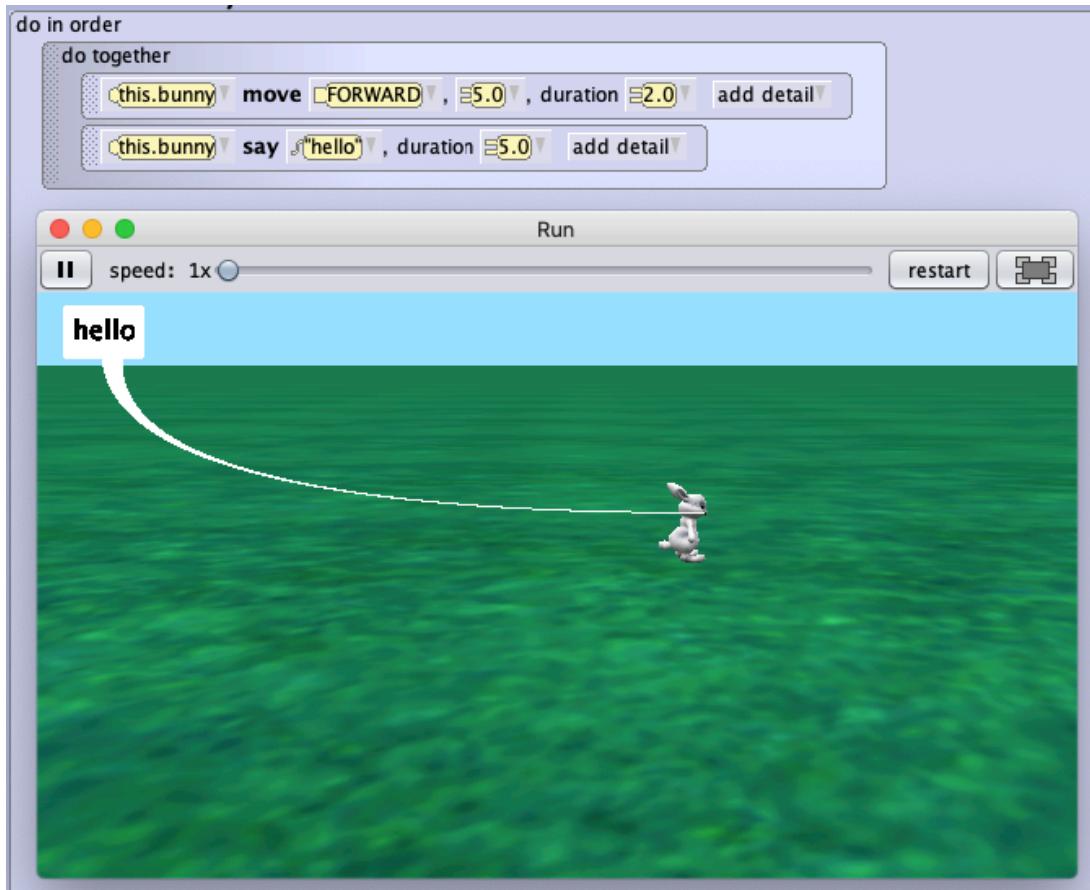
Procedure bipedJump dapat digunakan pada semua class biped, ini disebut sebagai pewarisan (inheritance)

27. Argumen pada Alice 3 dapat diubah sesuai dengan kebutuhan, adapun argumen tersebut terdiri dari object, direction, direction amount, time duration

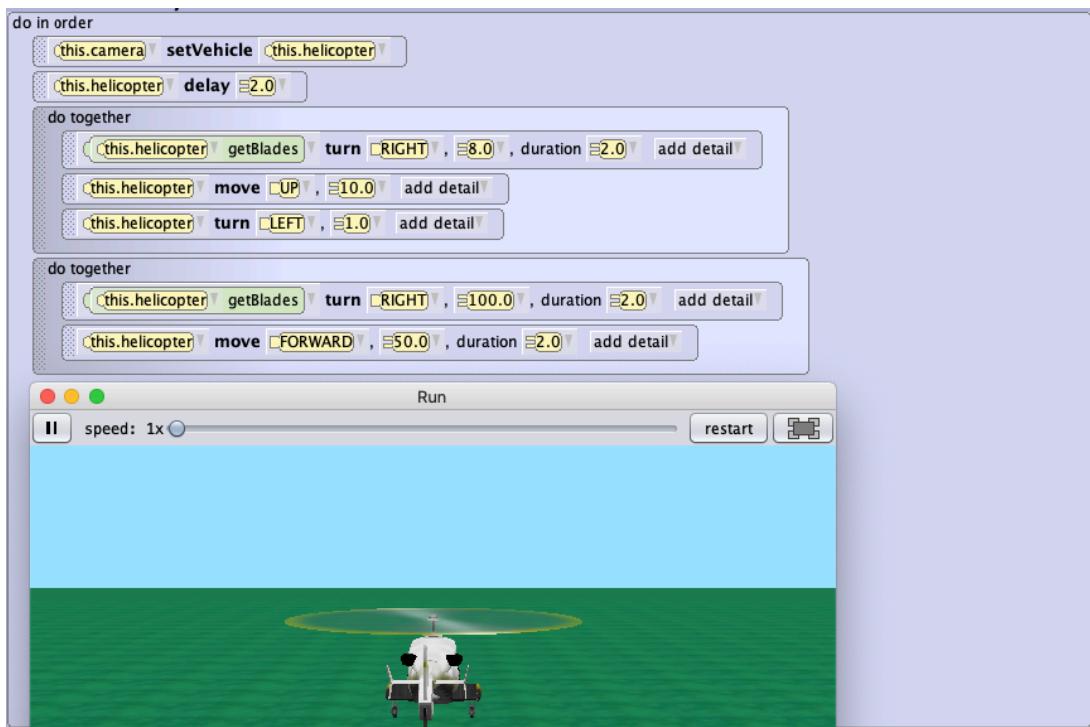


## 28. Control statement do in order dan do together

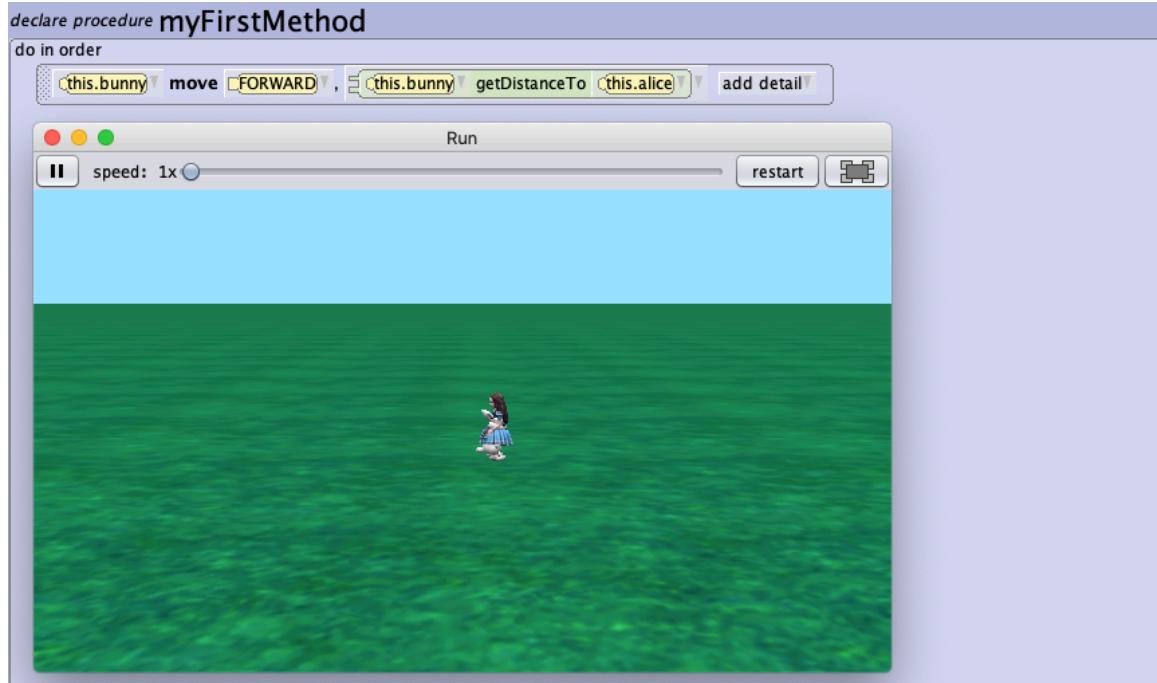




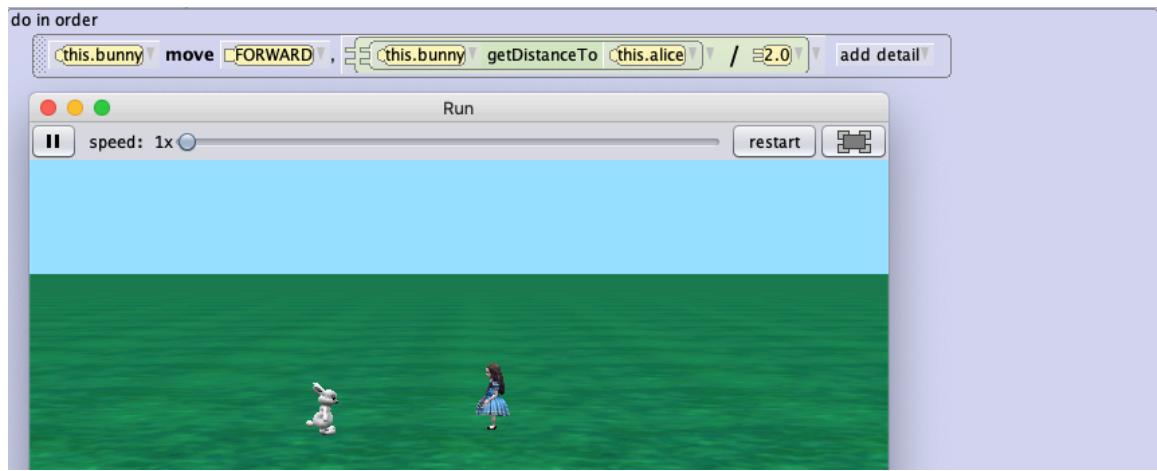
## 29. Menggunakan procedure setVehicle



30. Seperti yang sudah dijelaskan sebelumnya bahwa function adalah metode yang mengembalikan nilai. Pada Alice 3 function tidak dapat digunakan secara langsung, function hanya mengembalikan nilai. Berikut ini contoh penggunaan function.



31. Menambahkan operator pada function



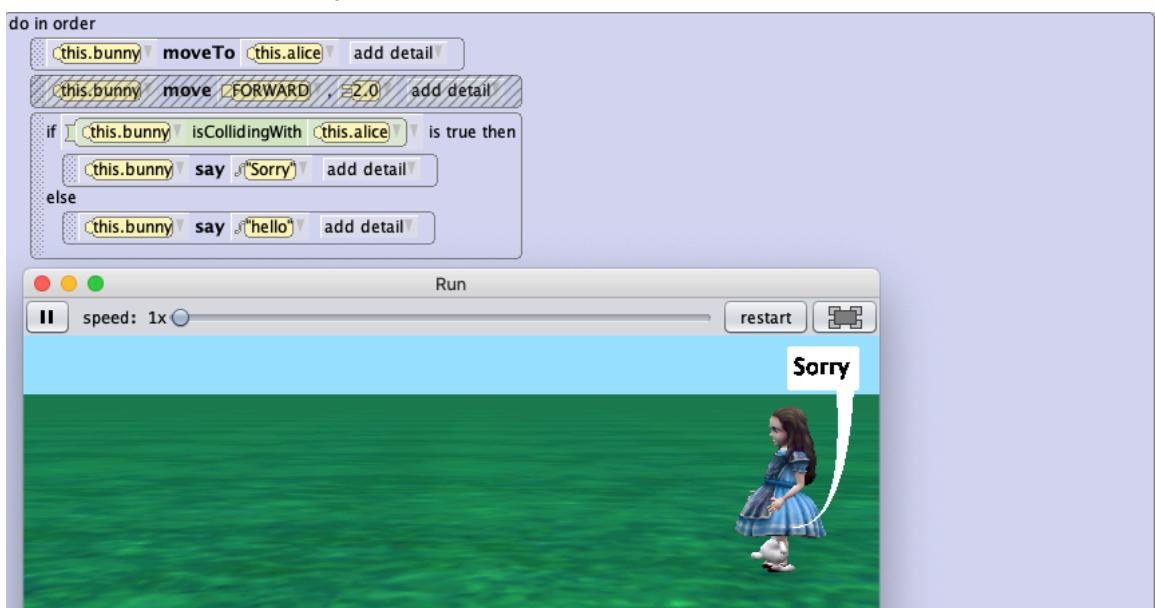
32. Menambahkan function lain untuk menagani operator sebelumnya



*nb:cek nilai depth pada setup scene*

### 33. Kuis

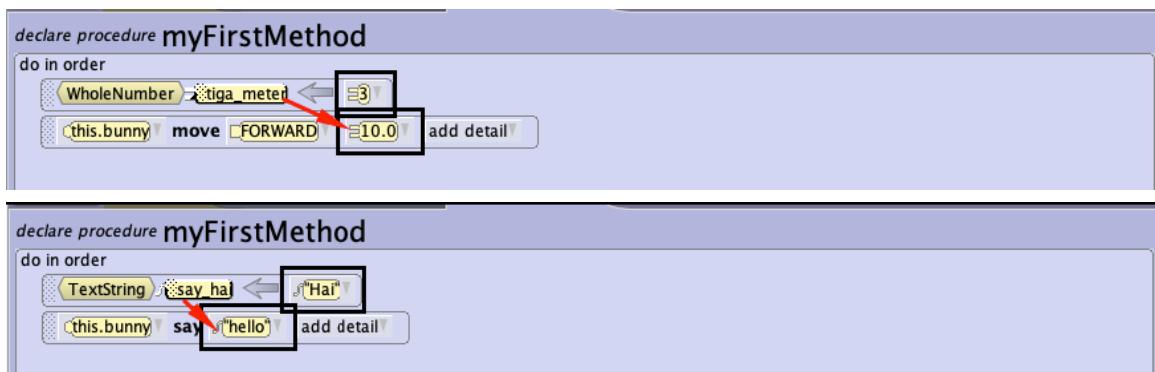
### 34. Kontrol statement IF pada Alice

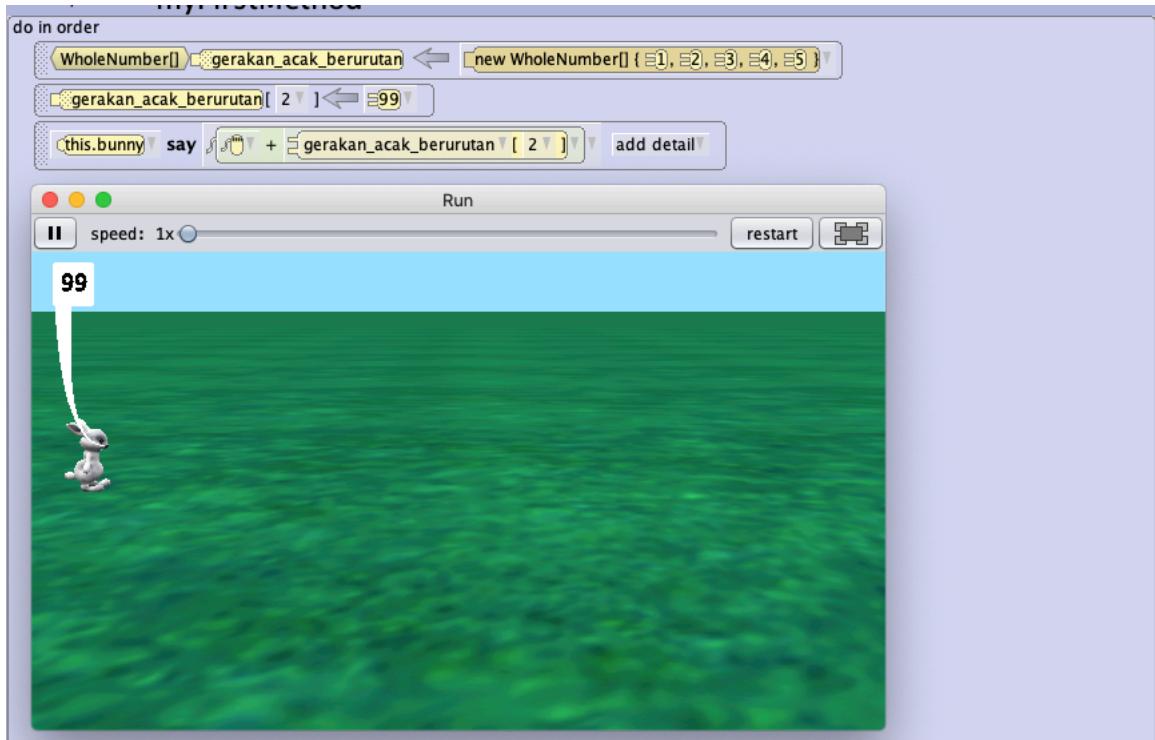


### 35. Kontrol statement WHILE pada Alice

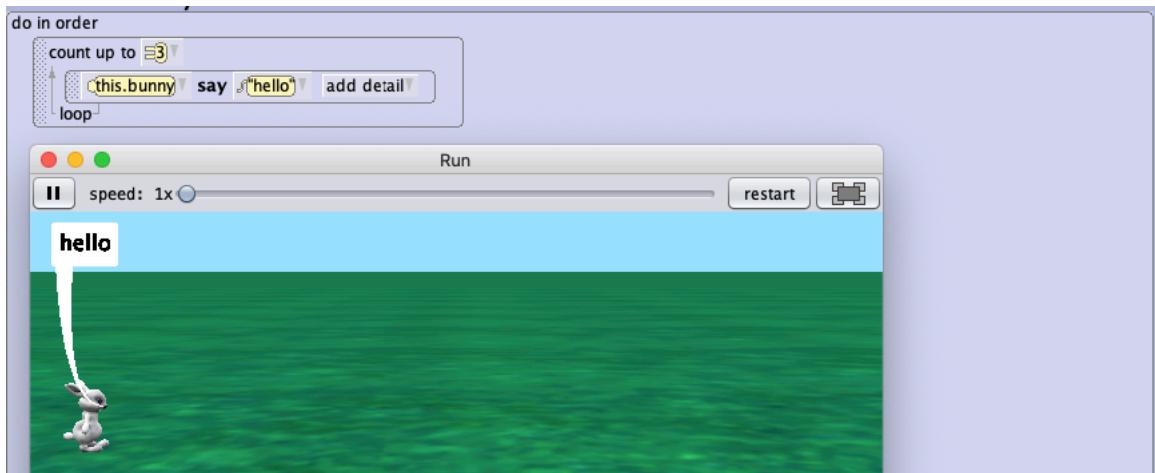


### 36. Variabel pada Alice (Cek tipe data pada slide JF 2 10)

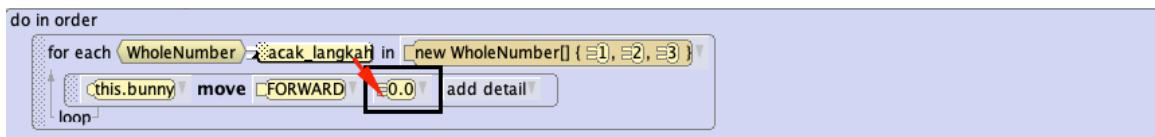




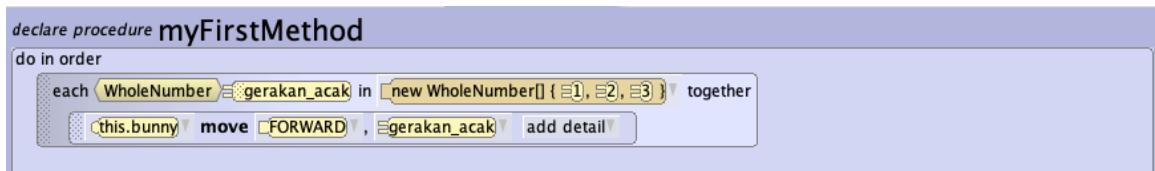
### 37. Kontrol statement COUNT



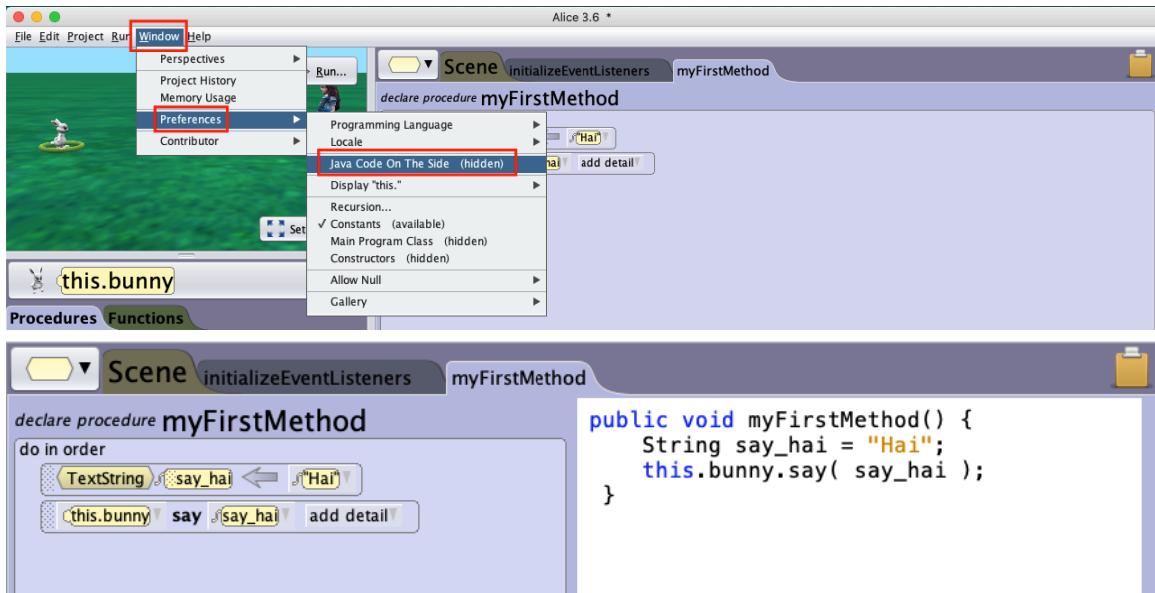
### 38. Kontrol Statement FOR EACH IN



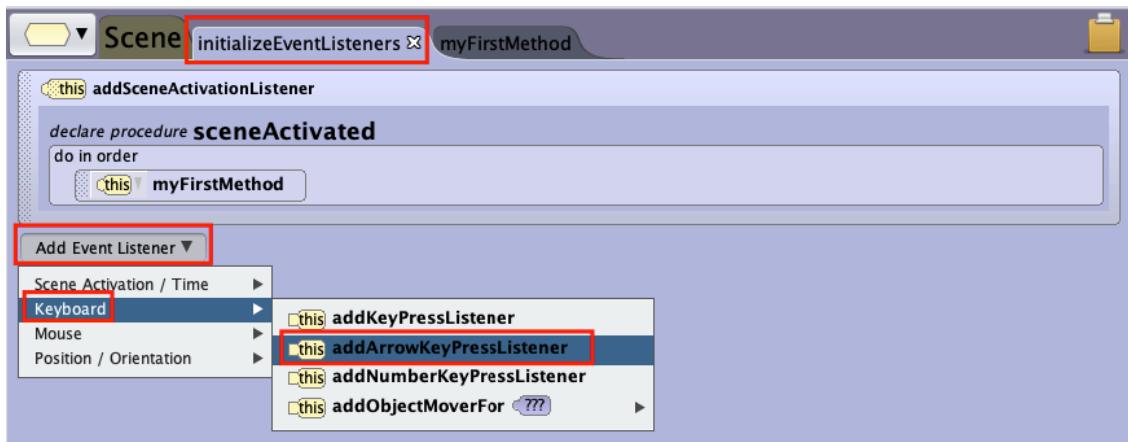
### 39. Kontrol Statement EACH IN TOGETHER

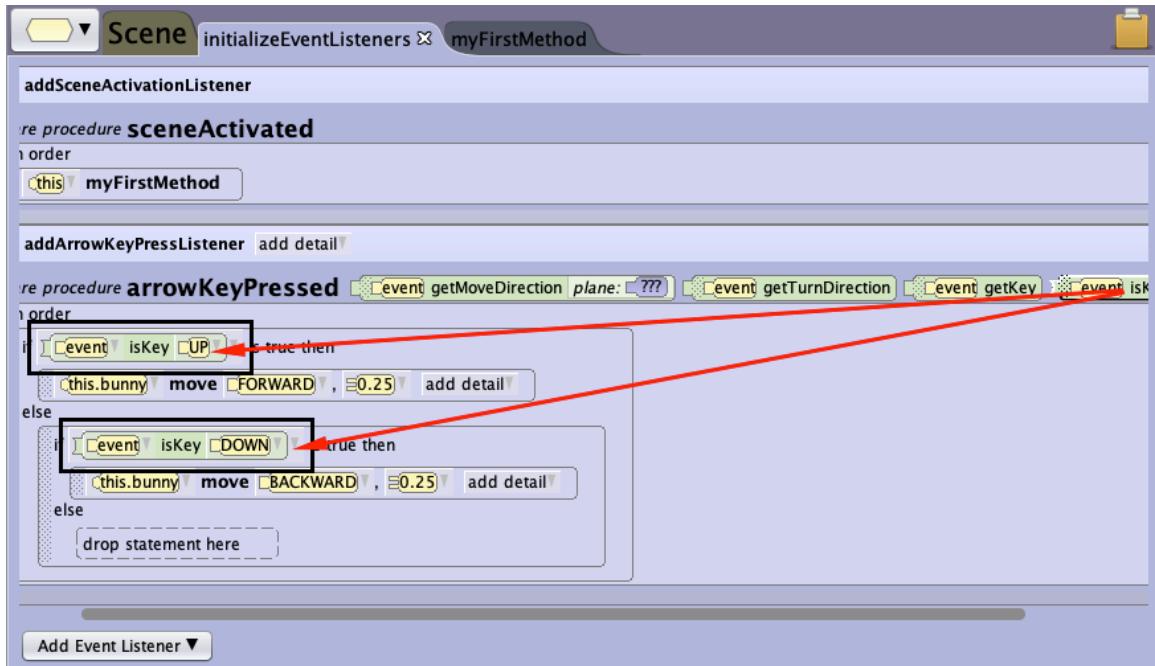


### 40. Melihat kode java pada Alice

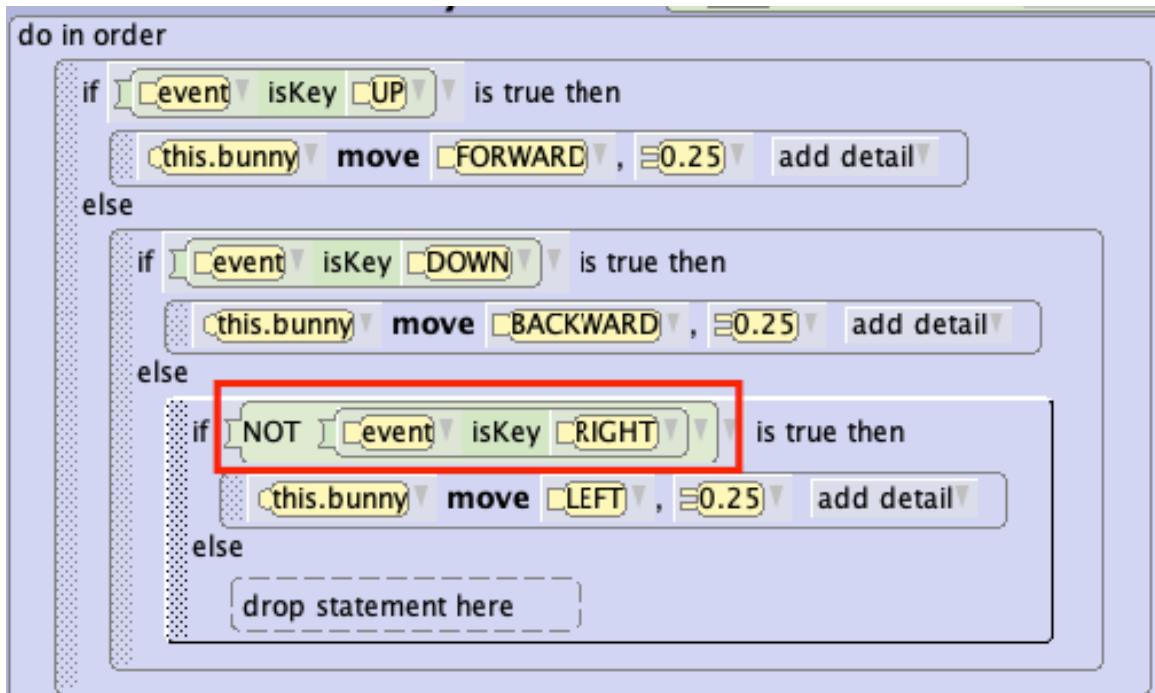


### 41. Kontrol melalui keyboard

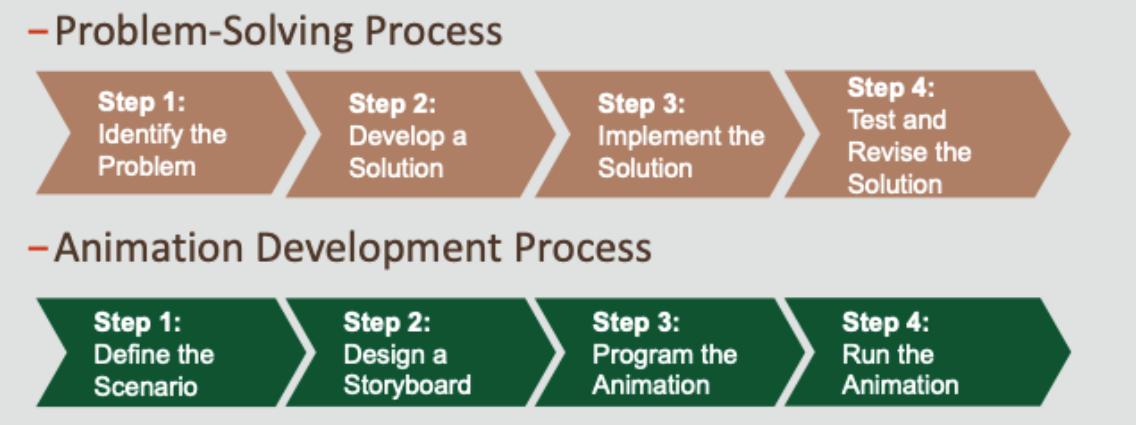




Bagaimana dengan ini!



42. Buatlah animasi lengkap dengan proses sebagai berikut!



43. Perbandingan variabel pada Alice dan Java

The screenshot shows a comparison between Alice script and Java code for a simple "Hello World" program.

**Alice Script:**

```
declare procedure myFirstMethod
do in order
    [TextString] hello_world ← ["Hello World"]
    [this.alice] say [hello_world] add detail
```

**Java Code:**

```
1 public class utama {
2     public static void main(String[] args) {
3         // TODO Auto-generated method stub
4         String hello_world = "Hello World";
5         System.out.println(hello_world);
6     }
7 }
```

44. Relational Operator

The screenshot shows Alice script using relational operators (`<`) within an `if`-`then`-`else` block.

**Alice Script:**

```
declare procedure myFirstMethod
do in order
    [this.bunny] move FORWARD, [this.bunny] getDistanceInFrontOf [this.alice] add detail add detail
    if [this.bunny] getHeight < [this.alice] getHeight is true then
        [this.alice] getHead turn FORWARD, 0.05 add detail
    else
        [drop statement here]
```

```
J utama.java X
1
2 public class utama {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         // Asumsi jarak antara bunny dan alice adalah 5 meter
7         double bunny = 0.59;
8         double alice = 1.42;
9
10        if (bunny < alice) {
11            System.out.println("Alice getHead Turn Forward 0.05");
12        }
13    }
14
15
16 }
17
```



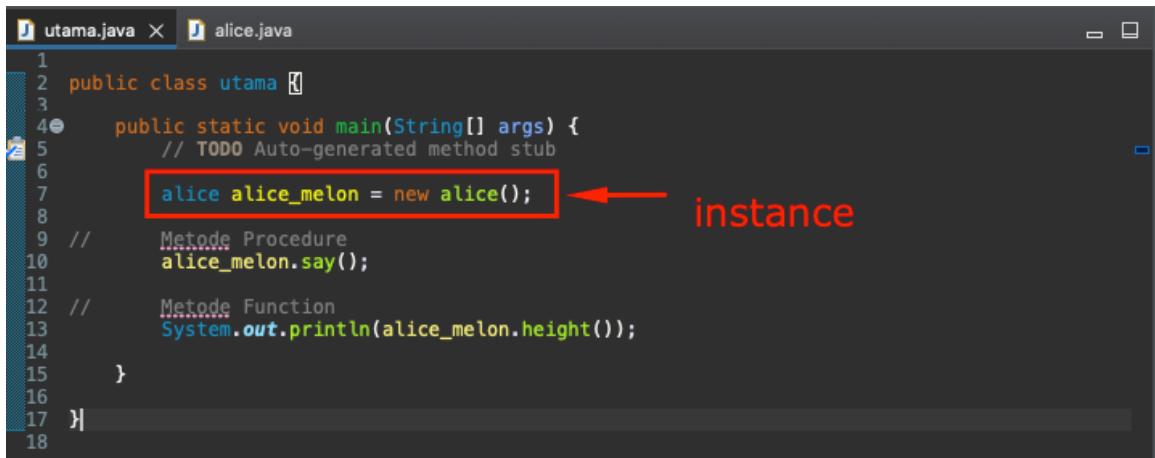
```
J utama.java X
1
2 public class utama {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         // Asumsi jarak antara bunny dan alice adalah 5 meter
7         String bunny = "behindAlice";
8         String alice = "behindBunny";
9
10        if ((bunny == "behindAlice") && (alice == "behindBunny")) {
11            System.out.println("Kelewatan");
12        }
13    }
14
15
16 }
17
```

#### 45. Assignment Operator



```
1 public class utama {
2
3
4    public static void main(String[] args) {
5        // TODO Auto-generated method stub
6        // Asumsi jarak antara bunny dan alice adalah 5 meter
7        int tiga_meter = 3;
8        int empat_meter = tiga_meter;
9
10       System.out.println(empat_meter);
11
12    }
13
14 }
15
```

#### 46. Mengenal class, objek dan instance pada Java



```
utama.java X alice.java
1
2 public class utama {
3
4    public static void main(String[] args) {
5        // TODO Auto-generated method stub
6
7        alice alice_melon = new alice(); ← instance
8
9        // Metode Procedure
10       alice_melon.say();
11
12      // Metode Function
13      System.out.println(alice_melon.height());
14
15    }
16
17 }
18
```

#### 47. Mengenal Metode Procedure dan Function pada Java



```
utama.java X alice.java
1
2 public class alice {
3
4    public void say() {
5        System.out.println("Hello");
6    }
7
8    public int height() {
9        return 1;
10    }
11
12
13 }
14
```

# PENGENALAN GREENFOOT

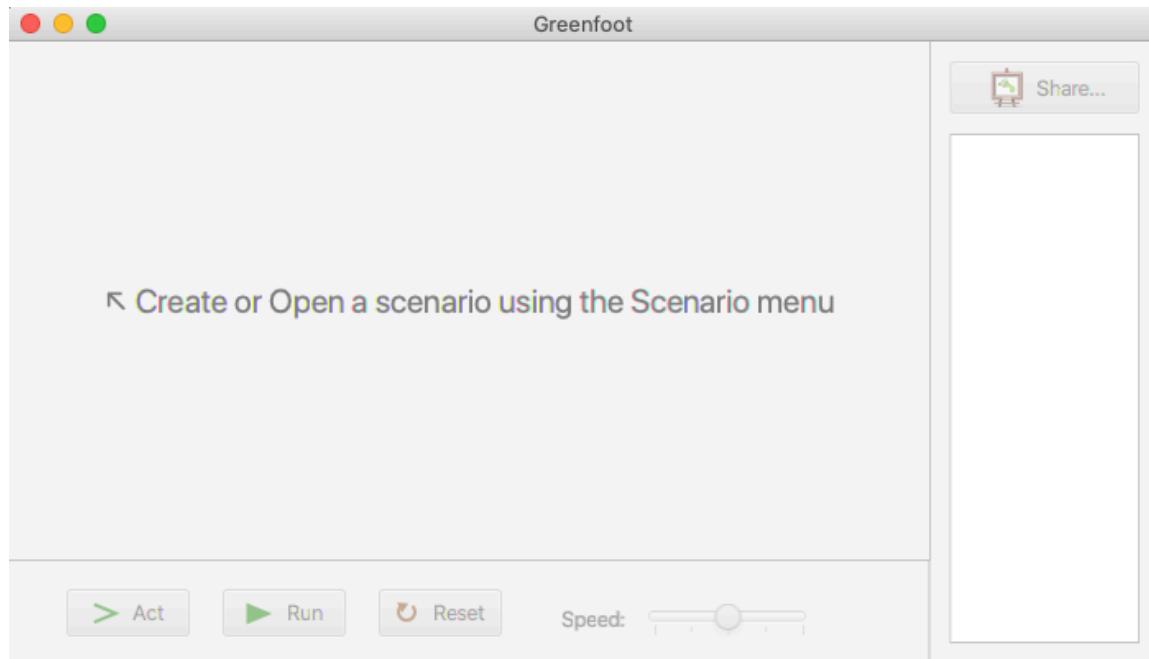
## Tujuan Pembelajaran:

1. Mengenal Greenfoot
2. Memulai dengan Greenfoot
3. Menggunakan metode, variabel dan parameter
4. Memanfaatkan source code dan documentation
5. Mengembangkan dan menguji aplikasi

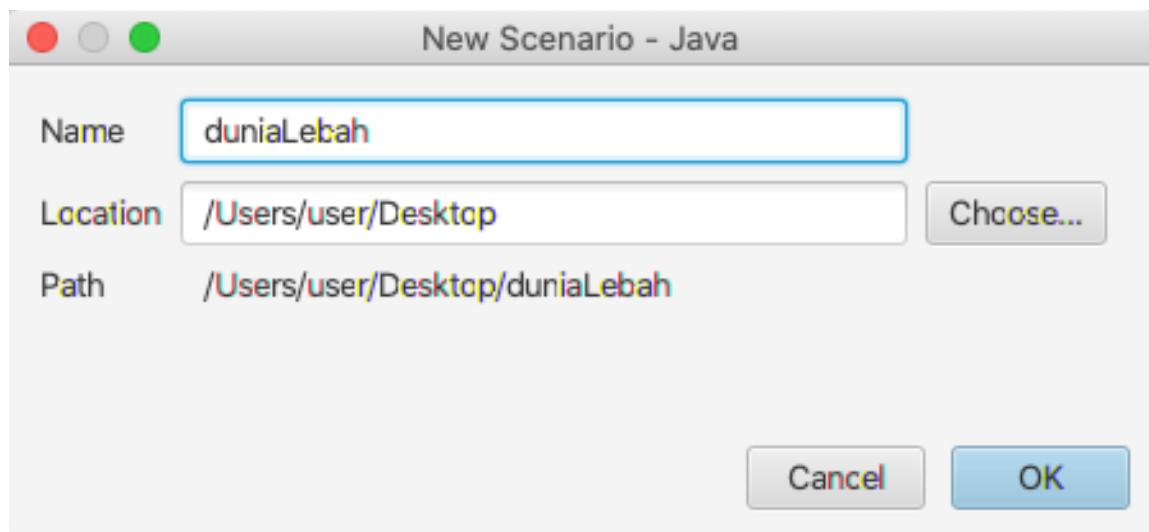
## Pengenalan Greenfoot

Greenfoot adalah perangkat lunak yang di desain untuk pemula agar terbiasa dengan pemrograman berorientasi objek (OOP), yang mendukung pengembangan aplikasi visual dengan menggunakan bahasa pemrograman java.

Memulai program greenfoot dapat dilakukan dengan membuka aplikasi greenfoot terlebih dahulu.



Membuat scenario dapat dilakukan dengan memilih menu scenario kemudian pilih new java scenario kemudian buat nama scenario.

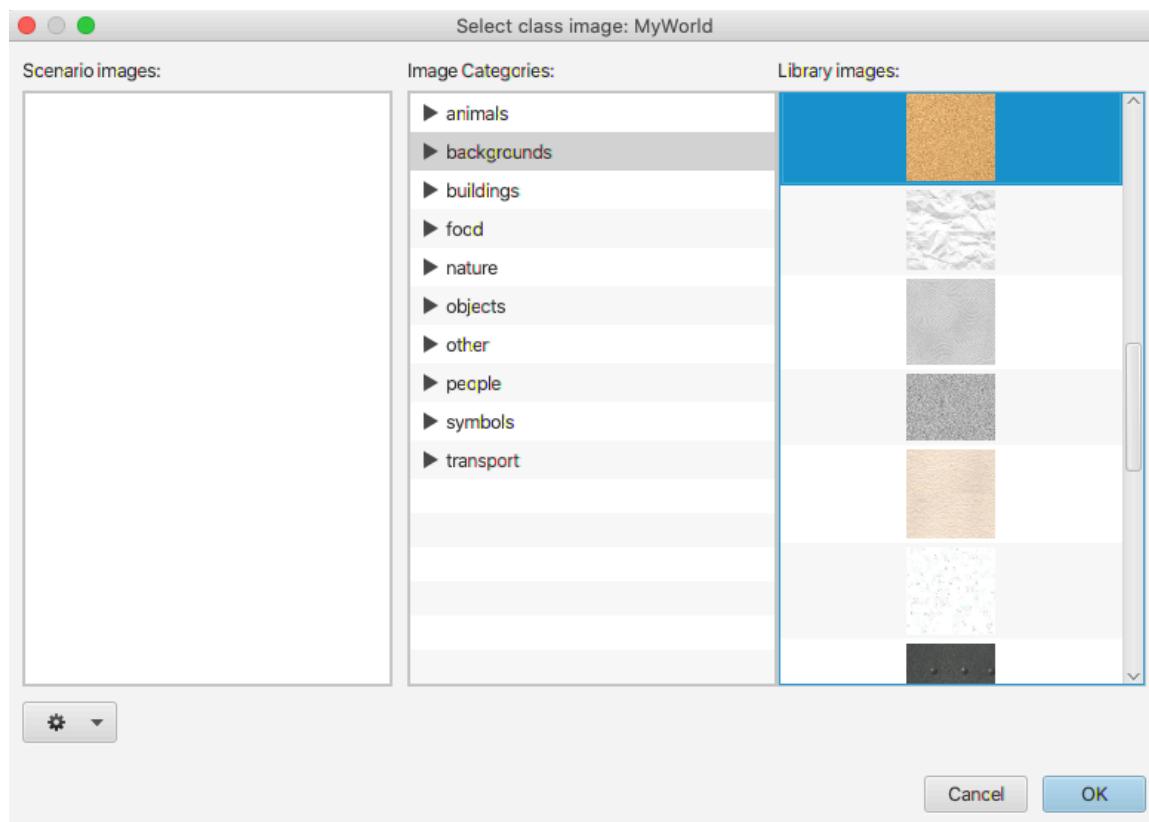


Jika sudah pilih OK kemudian berikut ini tampilan dari scenario yang telah dibuat.



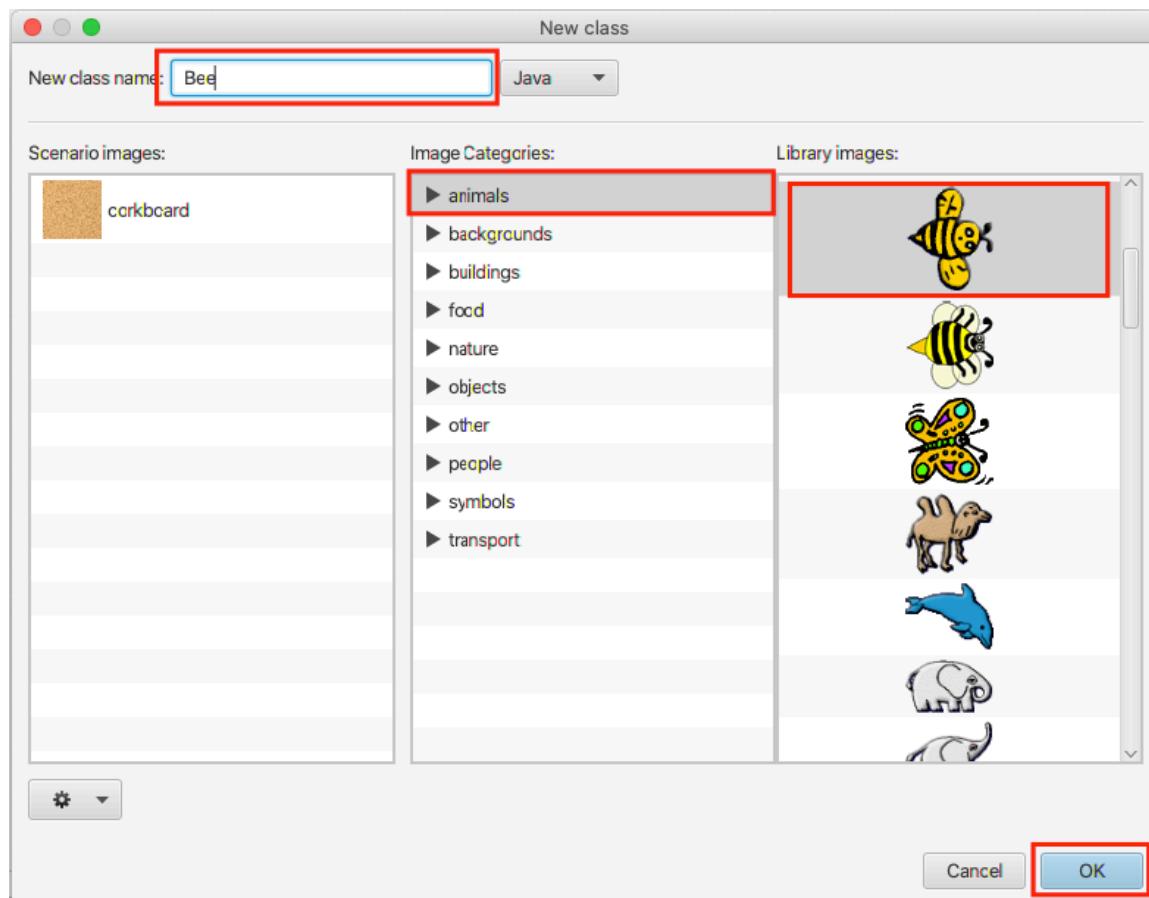
Pengguna dapat menambahkan gambar pada world yang telah tersedia dengan cara klik kanan pada MyWorld kemudian pilih set image,

selanjutnya pilih backgrounds. Pengguna dapat memilih latar belakang yang ingin digunakan.

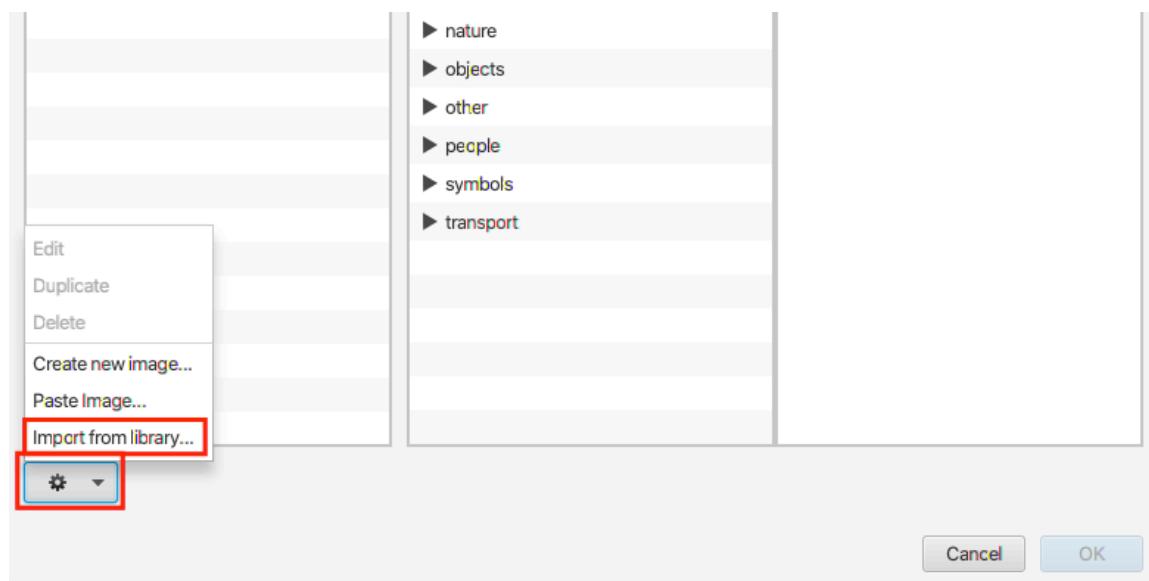


Jika sudah menentukan pilihan untuk latar belakang kemudian pilih OK. Agar latar belakang dapat diterapkan oleh world, klik kanan pada MyWorld kemudian pilih newMyWorld().

Greenfoot mengizinkan pengguna untuk menambahkan subclass melalui superclass yang sudah dipersiapkan. Misalkan saja menambah subclass baru dengan nama Bee pada superclass Actor dapat dilakukan dengan cara klik kanan pada superclass Actor kemudian pilih new subclass, pada jendela tambahkan nama class kemudian pilih gambar yang akan digunakan.



Selain menggunakan gambar yang tersedia, pengguna juga dapat menambahkan gambar lain dengan cara klik pada icon gear kemudian pilih import from library.

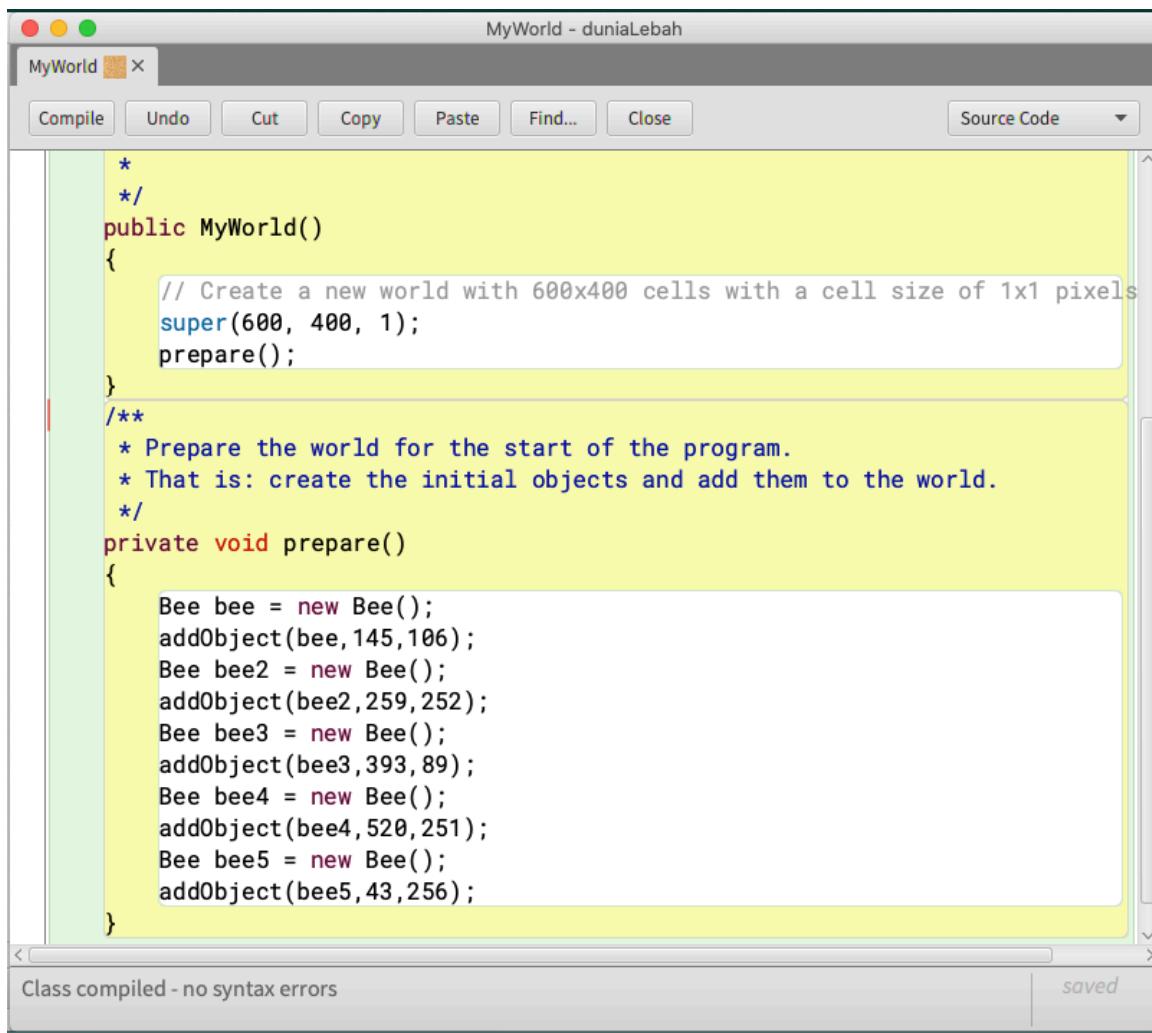


Scenario yang telah mengalami perubahan dapat disimpan dengan memilih scenario kemudian pilih save. Subclass Bee belum terlihat pada MyWorld, untuk dapat menampilkannya klik kanan pada subclass Bee kemudian pilih newBee() selanjutnya tempatkan pada MyWorld. Lakukan ujicoba dengan memilih tombol reset.



Semua Bee akan hilang atau kembali ke bentuk awal, agar Bee yang ditempatkan pada MyWorld tidak hilang dapat dengan tools kemudian pilih save the world. Lakukan ujicoba dengan memilih tombol reset, subclass Bee tidak akan hilang, perintah save the world juga secara otomatis menambahkan kode program pada MyWorld. Klik kanan pada subclass kemudian pilih open editor maka jendela source code untuk MyWorld akan terbuka.

Pada jendela source code terdapat beberapa control yang dapat digunakan untuk mengelola kode program, seperti compile untuk memeriksa kesalahan pada kode program, undo untuk kembali ke perubahan sebelumnya, cut untuk memindahkan (source code), copy untuk menempelkan (source code), find untuk mencari kata kunci dan close untuk menutup jendela source code.



```
/*
 */
public MyWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels
    super(600, 400, 1);
    prepare();
}

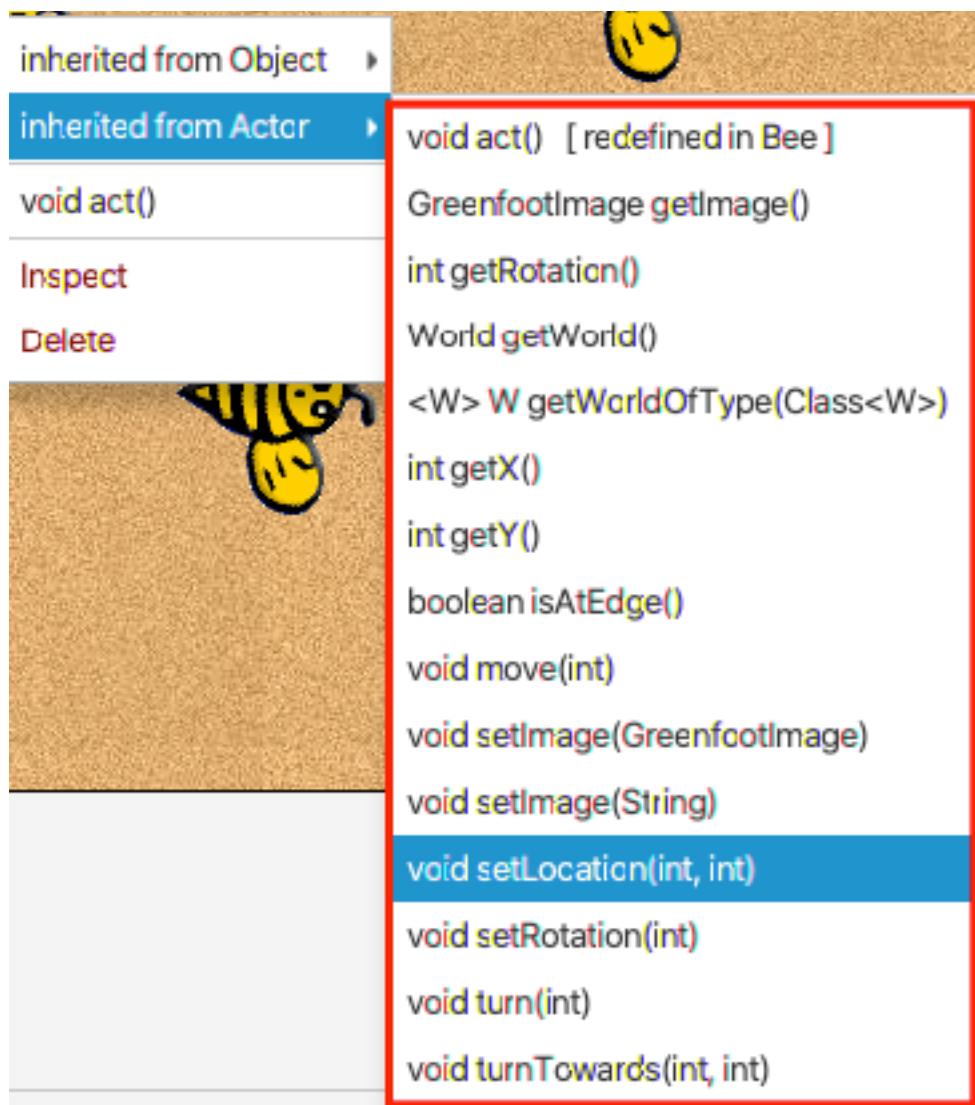
/**
 * Prepare the world for the start of the program.
 * That is: create the initial objects and add them to the world.
 */
private void prepare()
{
    Bee bee = new Bee();
    addObject(bee, 145, 106);
    Bee bee2 = new Bee();
    addObject(bee2, 259, 252);
    Bee bee3 = new Bee();
    addObject(bee3, 393, 89);
    Bee bee4 = new Bee();
    addObject(bee4, 520, 251);
    Bee bee5 = new Bee();
    addObject(bee5, 43, 256);
}
```

Class compiled - no syntax errors

Terdapat kontrol tambahan untuk melihat source code dan documentation pada sisi kanan atas jendela source code. Pada bagian bawah jendela source code digunakan untuk menampilkan pesan kesalahan atau error.

Dalam pemrograman juga terdapat istilah pewarisan atau inheritance, yang berarti metode yang ada pada superclass (parent) dapat diturunkan kepada subclass (child).

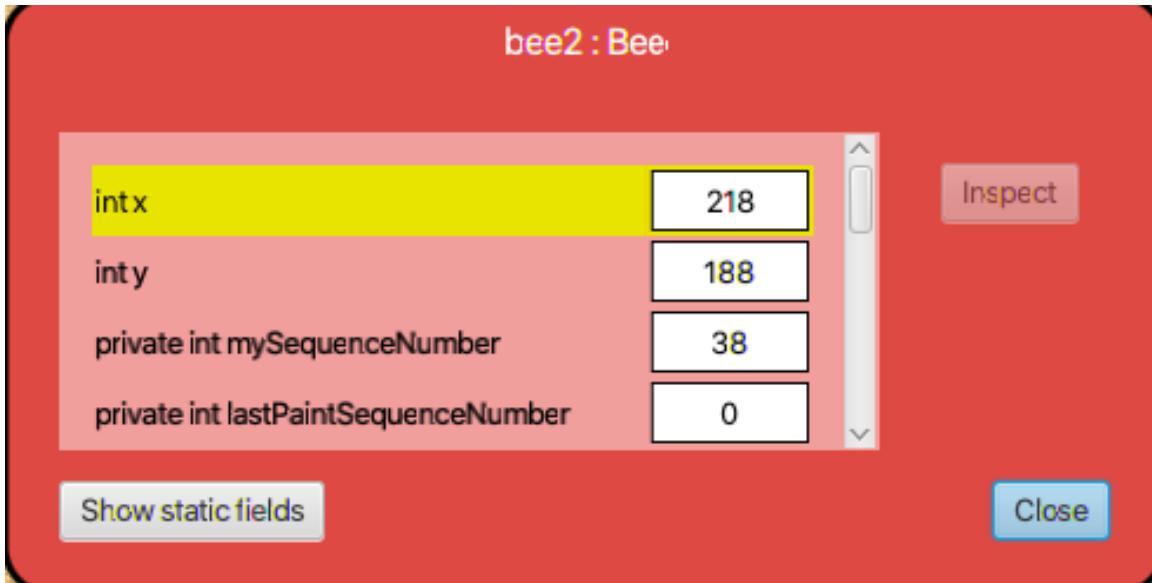
Pengguna dapat melihat metode yang diturunkan oleh superclass Actor kepada subclass Bee dengan cara klik kanan pada instance kemudian pilih inherited from actor. Terdapat cukup banyak metode yang diturunkan dari superclass. Metode tersebut juga terdiri dari procedure dan function.



Metode dalam pemrograman merupakan kumpulan operasi, metode dapat melakukan suatu tugas seperti berputar, berpindah atau metode yang secara khusus mendapatkan nilai. Beberapa komponen yang ada pada metode seperti tipe pengembalian (return type), nama metode dan parameter (contoh void move(3), int getX()), pada greenfoot juga terdapat metode dengan tipe pengembalian void dan metode dengan tipe pengembalian non-void (integer, boolean).

Pada pemrograman terdapat variabel yang digunakan untuk menampung data atau informasi yang nantinya akan digunakan kembali. Selanjutnya adalah object yang ditambahkan pada MyWorld memiliki properti yang disebut object properties. Object property ini

terdiri dari size, color, range of movement. Adapun cara yang dapat digunakan untuk melihatnya dengan cara klik kanan pada object kemudian pilih inspect.



## Metode act()

Metode act() merupakan metode yang akan dijalankan pertama kali dari semua metode yang ada. Tambahkan metode act() pada object hewan kemudian tambahkan metode berikut beserta parameternya.

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

Lakukan debugging untuk mengetahui kesalahan yang terjadi pada kode program.

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

Source code pada greenfoot merupakan blueprint atau peta yang mendefinisikan objek dan fungsi program, source code dapat di kelola melalui code editor, melalui code editor pengembang dapat menuliskan dan memodifikasi source code, meninjau metode pewarisan dan propertinya serta metode yang dituliskan secara spesifik.

Apa saja yang terdapat pada source code? Source code pada greenfoot terdiri dari class, metode act(), method signature (metode dengan nama metode dan parameternya, invoking method), komentar.

Greenfoot menyediakan dokumentasi yang bisa dijadikan sebagai acuan untuk belajar menggunakan metode-metode yang telah disediakan.

Metode sequential pada greenfoot berarti terdapat beberapa metode atau instruksi yang dituliskan secara urutan. Pada greenfoot juga terdapat statement if yang dapat digunakan untuk menggerakkan objek melalui keyboard.

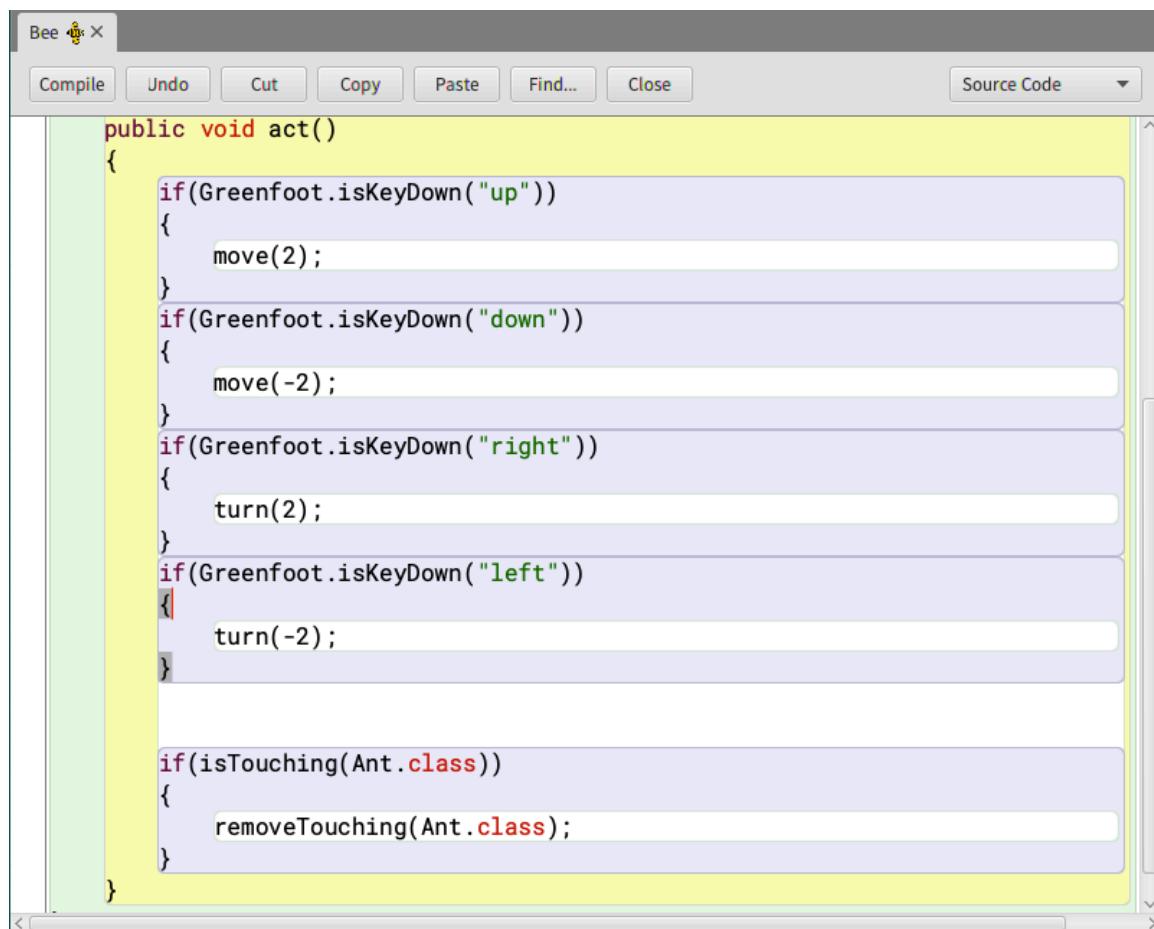
```
public void act()
{
    if(Greenfoot.isKeyDown("up"))
    {
        move(2);
    }
    if(Greenfoot.isKeyDown("down"))
    {
        move(-2);
    }
    if(Greenfoot.isKeyDown("right"))
    {
        turn(2);
    }
    if(Greenfoot.isKeyDown("left"))
    {
        turn(-2);
    }
}
```

Langkah dalam melakukan debugging program dapat dilakukan dengan compile, pengembang juga dapat melihat kesalahan yang terjadi dengan auto-layout (pada menu edit).

Terdapat beberapa fase yang dapat dilakukan untuk mengembangkan program dengan greenfoot, diantaranya:

1. Menganalisa permasalahan untuk dipecahkan
2. Merancang solusi
3. Mengembangkan game
4. Menguji game (sesuai dengan rancangan sebelumnya)
5. Evaluasi (Mengembangkan kembali dengan alur yang sama)

Membuat program sederhana dengan greenfoot:

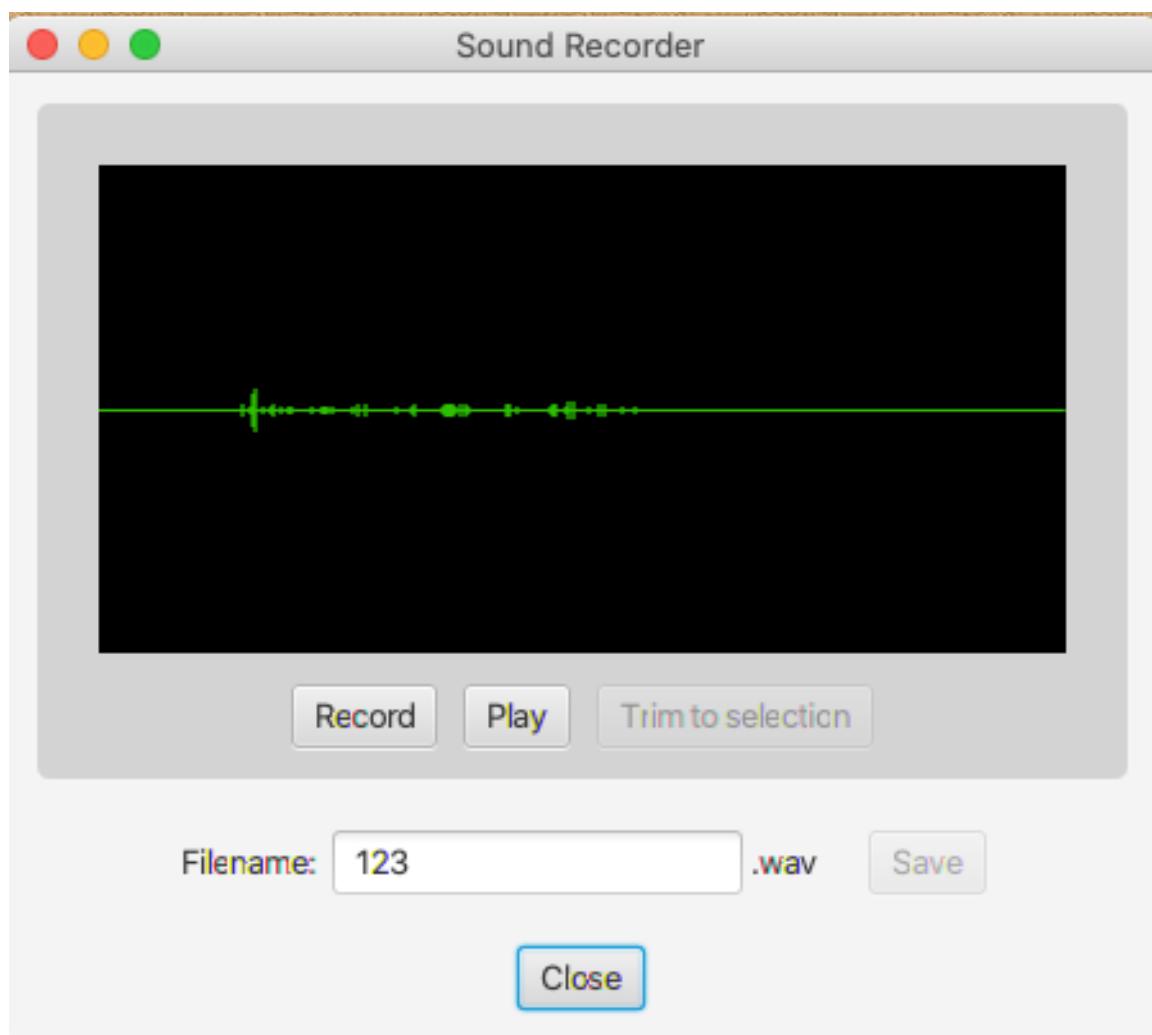


The screenshot shows the Greenfoot IDE interface with a script titled "Bee". The code in the editor is as follows:

```
public void act()
{
    if(Greenfoot.isKeyDown( "up" ))
    {
        move(2);
    }
    if(Greenfoot.isKeyDown( "down" ))
    {
        move(-2);
    }
    if(Greenfoot.isKeyDown( "right" ))
    {
        turn(2);
    }
    if(Greenfoot.isKeyDown( "left" ))
    {
        turn(-2);
    }

    if(isTouching(Ant.class))
    {
        removeTouching(Ant.class);
    }
}
```

Menambahkan suara pada permainan dapat dengan memilih menu tools kemudian pilih show sound recorder.



Lakukan perekaman suara dengan klik tombol record dan selesai merekam dengan klik tombol stop. Kemudian tambahkan nama file rekaman dan simpan rekaman. Agar dapat menggunakan rekaman tersebut tambahkan instruksi berikut.

```
if(isTouching(Ant.class))
{
    Greenfoot.playSound("123.wav");
    removeTouching(Ant.class);
}
```

Menambahkan nilai random dengan membuat salah satu objek melakukan rotasi dengan nilai yang diacak dapat dilakukan pada greenfoot dengan menambahkan perintah berikut.

```
public void act()
{
    // Add your action code here.
    move(10);
    if(isAtEdge())
    {
        turn(Greenfoot.getRandomNumber(90));
    }
}
```

getRandomNumber(90) berarti program akan mengacak angka dengan rentang antara 0 sampai dengan 89.

Pada greenfoot juga dapat digunakan operator pembanding, berikut ini contoh penggunaan operator pembanding pada greenfoot.

```
public void act()
{
    // Add your action code here.
    if(Greenfoot.getRandomNumber(100) < 20)
    {
        move(10);
        turn(20);
    }
}
```

Instruksi di atas menunjukan bahwa jika bilangan random kurang dari 20, maka akan ada perpindahan posisi dan juga rotasi yang terjadi pada objek. Selain operator pembanding juga terdapat operator logika seperti berikut ini.

```
public void act()
{
    // Add your action code here.
    if((getX() > getWorld().getWidth()/2) &&
    (getY() < getWorld().getHeight()/2))
    {
        move(10);
    }
}
```

Pemrogram juga dapat menambahkan logikan or (||) atau menambahkan not (!). Greenfoot juga mendukung penggunaan percabangan dalam instruksiannya.

```
public void act()
{
    // Add your action code here.
    move(10);
    if(Greenfoot.getRandomNumber(90) < 20)
    {
        turn(90);
    }
    else
    {
        move(5);
    }
}
```

Sama halnya dengan java, greenfoot juga memiliki konstruktor pada class myWorld, adapun bentuk konstruktornya adalah sebagai berikut.

```
public class MyWorld extends World
{
    /**
     * Constructor for objects of class MyWorld.
     *
     */
    public MyWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1
        super(800, 600, 1);
        prepare();
    }
}
```

Konstruktor adalah sebuah metode yang namanya mirip dengan nama class tempat metode tersebut berada. Pemrogram dapat menambahkan objek secara manual ke world yang tersedia dengan menambahkan instruksi berikut ini pada metode MyWorld.

```
public MyWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1
    super(800, 600, 1);
    prepare();
    addObject(new Ant(), 400, 300);
}
```

Penggunaan kata kunci new menunjukkan adanya instansiasi class, sehingga class blueprint (class object) dapat digunakan.

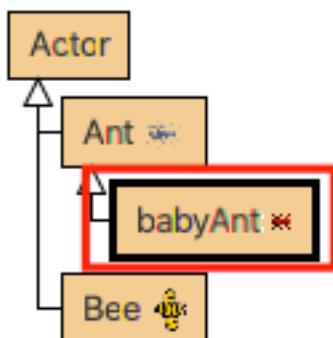
Pemrogramm dapat membuka dokumentasi untuk melihat beberapa metode built-in yang dapat digunakan untuk merekayasa animasi, misalkan saja menggunakan salah satu metode yaitu setLocation.

```

public void act()
{
    // Add your action code here.
    move(10);
    if(isAtEdge())
    {
        setLocation(200, 300);
    }
}

```

Menjadikan subclass sebagai superclass, dapat dilakukan pada greenfoot dengan menambahkan class seperti biasa.



Menambahkan konsep inheritance pada greenfoot dapat dilakukan dengan menambahkan sebuah metode procedure pada class Ant kemudian nanti metode ini dapat digunakan oleh babyAnt.

Tambahkan kode berikut pada class Ant:

```

public void teleportasi()
{
    move(10);
    if(isAtEdge())
    {
        setLocation(Greenfoot.getRandomNumber(200),
                    Greenfoot.getRandomNumber(200));
    }
}

```

Kemudian panggil metode tersebut pada class babyAnt:

```
public void act()
{
    // Add your action code here.
    teleportasi();
}
```

Memberikan batasan pada tepi kanan pada objek dapat dilakukan dengan menambahkan metode fungsi pada superclass Ant.

```
public boolean atRightEdge()
{
    if(getX() > getWorld().getWidth() - 20)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Panggil metode atRightEdge tersebut pada class babyAnt:

```
public void act()
{
    // Add your action code here.
    move(10);
    if(atRightEdge())
    {
        setLocation(400, 300);
    }
}
```

Metode ask pada greenfoot memberikan inputan dari pengguna, ini dapat dilakukan dengan menambahkan instruksi berikut ini.

```
String nama = Greenfoot.ask("Nama Anda: ");
int n = Integer.parseInt(nama);
if(Greenfoot.isKeyDown("UP"))
{
    move(n);
}
```

Menjadikan instruksi bergerak dengan keyboard dan gerakan menghilangkan object sebagai metode.

```
public void gerakanBee()
{
    if(Greenfoot.isKeyDown("UP"))
    {
        move(5);
    }

    if(Greenfoot.isKeyDown("DOWN"))
    {
        move(-5);
    }

    if(Greenfoot.isKeyDown("RIGHT"))
    {
        turn(5);
    }

    if(Greenfoot.isKeyDown("LEFT"))
    {
        turn(-5);
    }
}
```

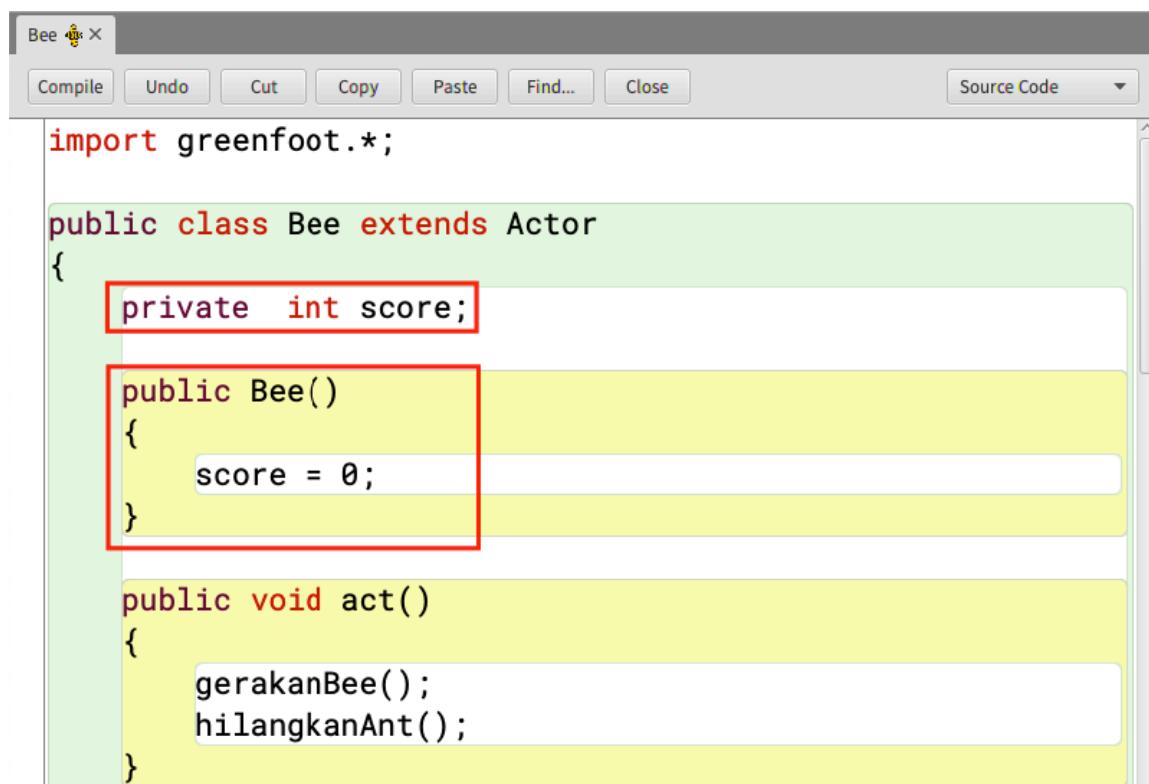
```
public void hilangkanAnt()
{
    if(isTouching(Ant.class))
    {
        //Greenfoot.playSound("hai.wav");
        removeTouching(Ant.class);
    }
}
```

Lakukan pemanggilan nama metode pada class utama (class act) agar dapat menggunakan instruksi.

```
public void act()
{
    gerakanBee();
    hilangkanAnt();
}
```

Abstraction pada greenfoot pada kasus ini menggunakan contoh pembuatan skor permainan.

Tambahkan property pada class Bee dan juga konstruktor seperti berikut ini:



The screenshot shows the Greenfoot code editor with the class "Bee" selected. The code is as follows:

```
import greenfoot.*;

public class Bee extends Actor
{
    private int score;

    public Bee()
    {
        score = 0;
    }

    public void act()
    {
        gerakanBee();
        hilangkanAnt();
    }
}
```

A red box highlights the constructor definition `public Bee()` and its body, while the variable declaration `private int score;` is highlighted with a red border.

Kemudian tambahkan instruksi atau perintah berikut pada metode hilangkan Ant.

```
public void hilangkanAnt()
{
    if(isTouching(Ant.class))
    {
        //Greenfoot.playSound("hai.wav");
        removeTouching(Ant.class);
        score = score + 20;
        getWorld().showText("Score: "+score,80,25);
    }
}
```

Agar instruksi ini dapat digunakan berkali-kali perlu dipisahkan menjadi sebuah metode, ini ditujukan untuk menghilangkan penggunaan kode secara berulang, sehingga cukup memanggil metodenya saja.

Tambahkan sebuah metode baru dengan instruksi yang sama dengan sebelumnya.

```
private void addScore()
{
    score = score + 20;
}
```

Kemudian lakukan pemanggilan metode addScore pada metode hilangkanAnt.

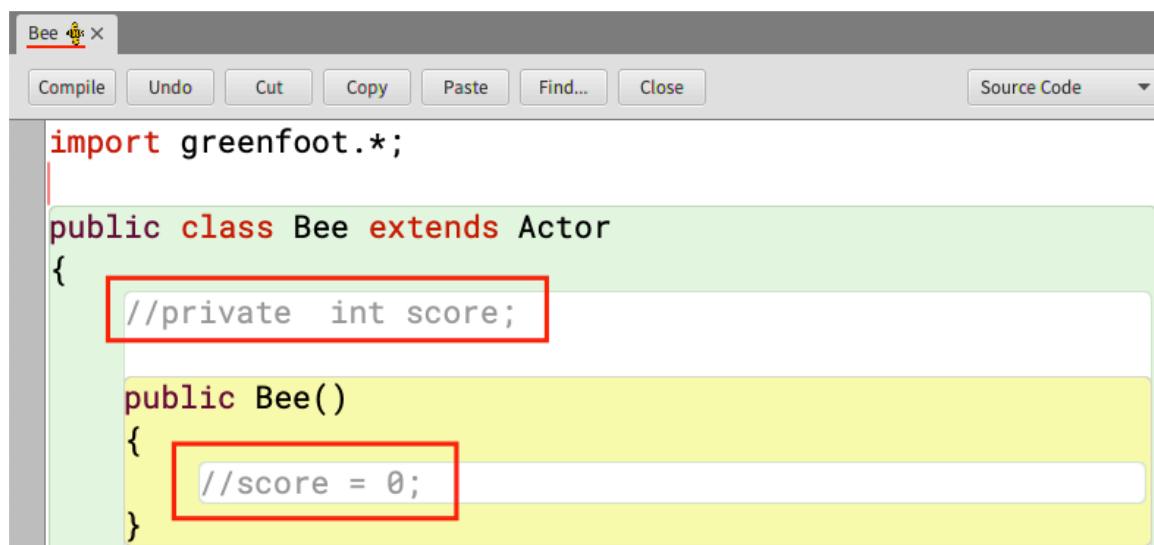
```
public void hilangkanAnt()
{
    if(isTouching(Ant.class))
    {
        //Greenfoot.playSound("hai.wav");
        removeTouching(Ant.class);
        //score = score + 20;
        addScore();
        getWorld().showText("Score: "+score,80,25);
    }
}
```

Instruksi ini masih dapat dimodifikasi dengan menambahkan parameter pada metode addScore dan pada proses pemanggilan metode addScore ditambahkan argument.

```
public void hilangkanAnt()
{
    if(isTouching(Ant.class))
    {
        //Greenfoot.playSound("hai.wav");
        removeTouching(Ant.class);
        //score = score + 20;
        addScore(20);
        getWorld().showText("Score: "+score, 80, 25);
    }
}

private void addScore(int point)
{
    score = score + point;
}
```

Selanjutnya ada casting, untuk dapat menggunakan casting pada greenfoot lakukan penghapusan pada metode addScore, property private int score dan juga score = 0 pada konstruktor serta .



The screenshot shows the Greenfoot IDE interface with the title bar "Bee". Below the title bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", "Close", and "Source Code". The main code editor area contains the following Java code:

```
import greenfoot.*;

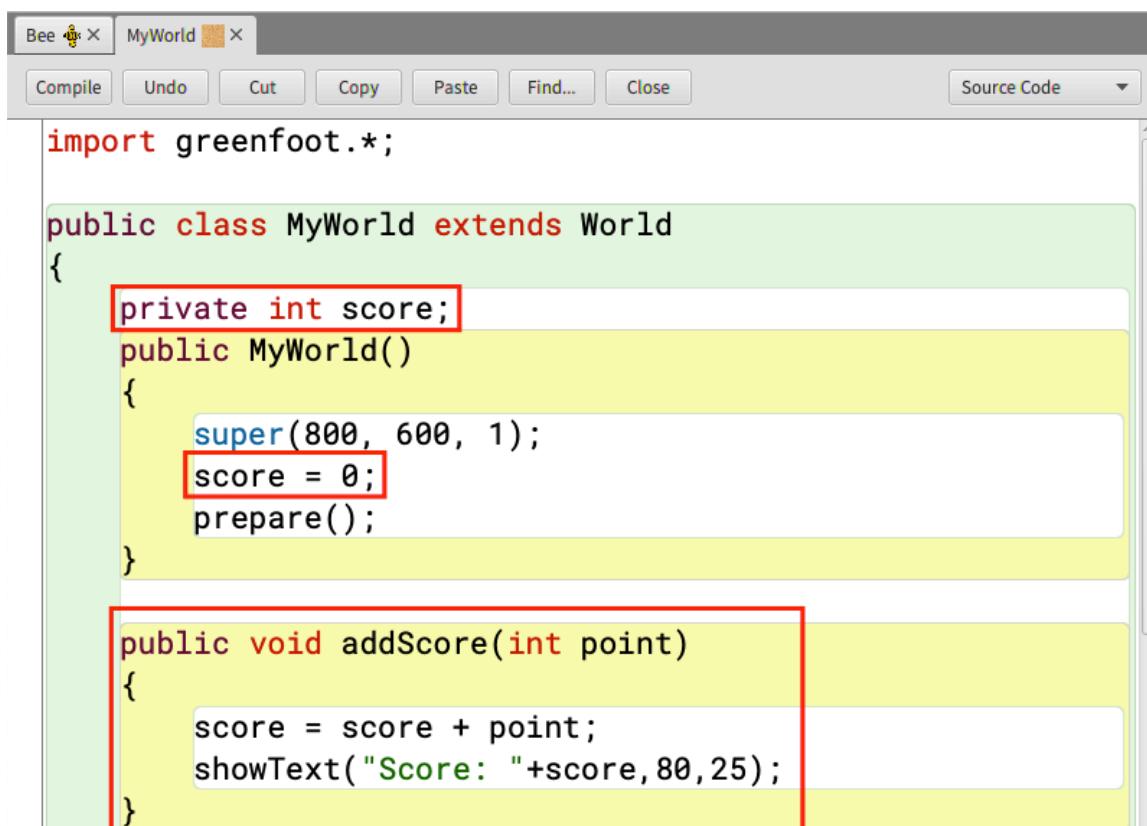
public class Bee extends Actor
{
    //private int score;

    public Bee()
    {
        //score = 0;
    }
}
```

The line `//private int score;` is highlighted with a red rectangle. The line `//score = 0;` is also highlighted with a red rectangle.

```
/**private void addScore(int point)
{
    score = score + point;
}*/
```

Selanjutnya tambahkan perintah berikut pada class MyWorld:



The screenshot shows the Greenfoot IDE interface with two tabs open: 'Bee' and 'MyWorld'. The 'MyWorld' tab is active, displaying the following Java code:

```
import greenfoot.*;

public class MyWorld extends World
{
    private int score;
    public MyWorld()
    {
        super(800, 600, 1);
        score = 0;
        prepare();
    }

    public void addScore(int point)
    {
        score = score + point;
        showText("Score: "+score, 80, 25);
    }
}
```

The code is color-coded: 'import' and 'public' are red, 'MyWorld' and 'World' are blue, and variable names like 'score' and method parameters like 'point' are black. The declaration of 'score' and its assignment in the constructor, as well as the entire definition of the 'addScore' method, are highlighted with a red rectangular box.

Berikutnya tambahkan instruksi untuk melakukan casting pada metode hilangkanAnt di dalam class Bee seperti berikut:

```
public void hilangkanAnt()
{
    if(isTouching(Ant.class))
    {
        //Greenfoot.playSound("hai.wav");
        removeTouching(Ant.class);
        //score = score + 20;
        //addScore(20);
        //getWorld().showText("Score: "+score,80,25);
        MyWorld myworld = (MyWorld)getWorld();
        myworld.addScore(20);
    }
}
```

Contoh penerapan perulangan pada Greenfoot dapat dilakukan dengan menambahkan instruksi berikut pada class world.

```
private void prepare()
{
    Bee bee = new Bee();
    addObject(bee,600,200);

    int i = 0;
    while(i < 100)
    {
        int koor_x = Greenfoot.getRandomNumber(400);
        int koor_y = Greenfoot.getRandomNumber(600);
        addObject(new Ant(), koor_x, koor_y);
        i = i + 1;
    }
}
```

Membuat variabel pada Greenfoot seperti berikut ini:

```
import greenfoot.*;  
  
public class Bee extends Actor  
{  
    private GreenfootImage imageright;  
    private GreenfootImage imageleft;  
    //private int score;  
    boolean isTurning = false;  
  
    public Bee()  
    {  
        imageright = new GreenfootImage("bee_right.png");  
        imageleft = new GreenfootImage("bee_left.png");  
    }  
}
```

The code editor window shows the 'Bee' class definition. A red box highlights the declaration of two private variables: 'imageright' and 'imageleft'. Another red box highlights the constructor 'Bee()' which initializes these variables to images named 'bee\_right.png' and 'bee\_left.png' respectively.

\*import gambarnya terlebih dahulu

```
if(Greenfoot.isKeyDown("RIGHT"))  
{  
    turn(5);  
    isTurning = true;  
    setImage(imageleft);  
}  
  
if(Greenfoot.isKeyDown("LEFT"))  
{  
    turn(-5);  
    isTurning = true;  
    setImage(imageright);  
}
```

The code editor window shows the logic for handling keyboard events. It checks if the right key is down, turns the bee 5 degrees right, sets the 'isTurning' flag to true, and switches to the left image. It does the same for the left key, turning 5 degrees left, setting the 'isTurning' flag to true, and switching to the right image.

Menggunakan array pada greenfoot dapat dilakukan dengan membuat variabel array dan menambahkan instruksi pada metode konstruktor seperti berikut ini:

```
import greenfoot.*;  
  
public class Bee extends Actor  
{  
    private GreenfootImage[] images = new GreenfootImage[4];  
    private int current_image;  
  
    public Bee()  
    {  
        //images[0] = new GreenfootImage("bee1.png");  
        //images[1] = new GreenfootImage("bee2.png");  
        //images[2] = new GreenfootImage("bee3.png");  
        //images[3] = new GreenfootImage("bee4.png");  
        //ubah menjadi seperti ini  
        int i = 0;  
        while(i < 4)  
        {  
            images[i] = new GreenfootImage("bee"+(i+1)+".png");  
            i = i+1;  
        }  
        current_image = 0;  
    }  
}
```

Kemudian tambahkan metode untuk membuat animasi gerakan sayap (terbang) pada Bee seperti berikut ini:

```
private void animasiBee()  
{  
    if(current_image == 3)  
    {  
        current_image = 0;  
    }  
    else  
    {  
        current_image++;  
    }  
    setImage(images[current_image]);  
}  
  
public void act()  
{  
    animasiBee();  
    if(Greenfoot.isKeyDown("UP"))  
    {  
        move(5);  
    }  
  
    if(Greenfoot.isKeyDown("DOWN"))  
    {  
        move(-5);  
    }  
}
```

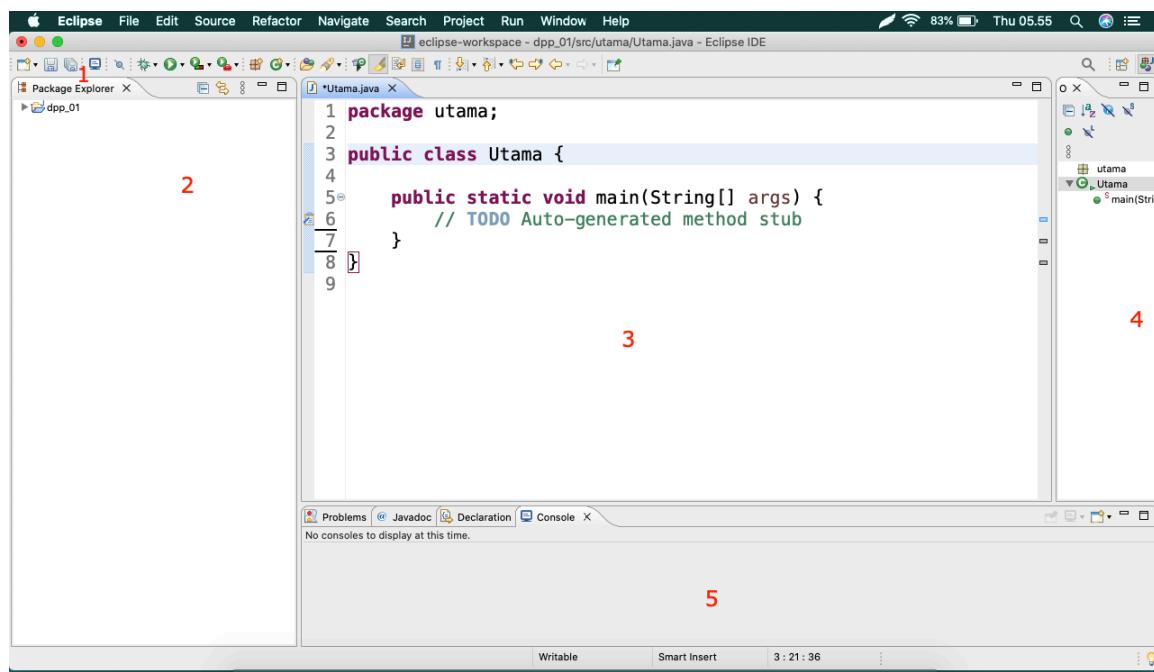
# PENGENALAN ECLIPSE

## Tujuan Pembelajaran:

1. Mengenal Eclipse
2. Memahami Class Driver dan Class Object
3. Memahami Tipe Data dan Operator
4. Memahami Manipulasi String

## Pengenalan Eclipse

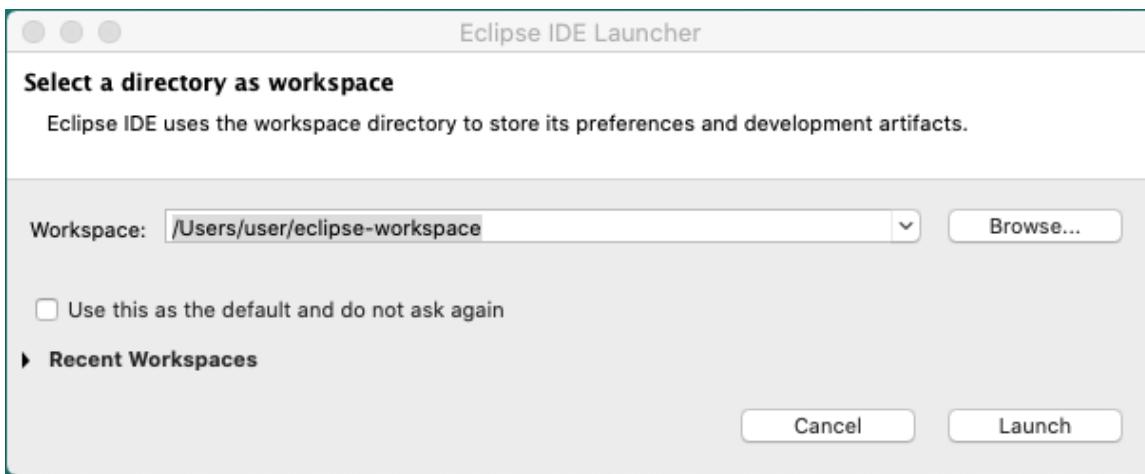
Eclipse merupakan Integrated Development Environment (IDE) yang digunakan untuk mengembangkan program aplikasi dan dapat dijalankan disemua platform.



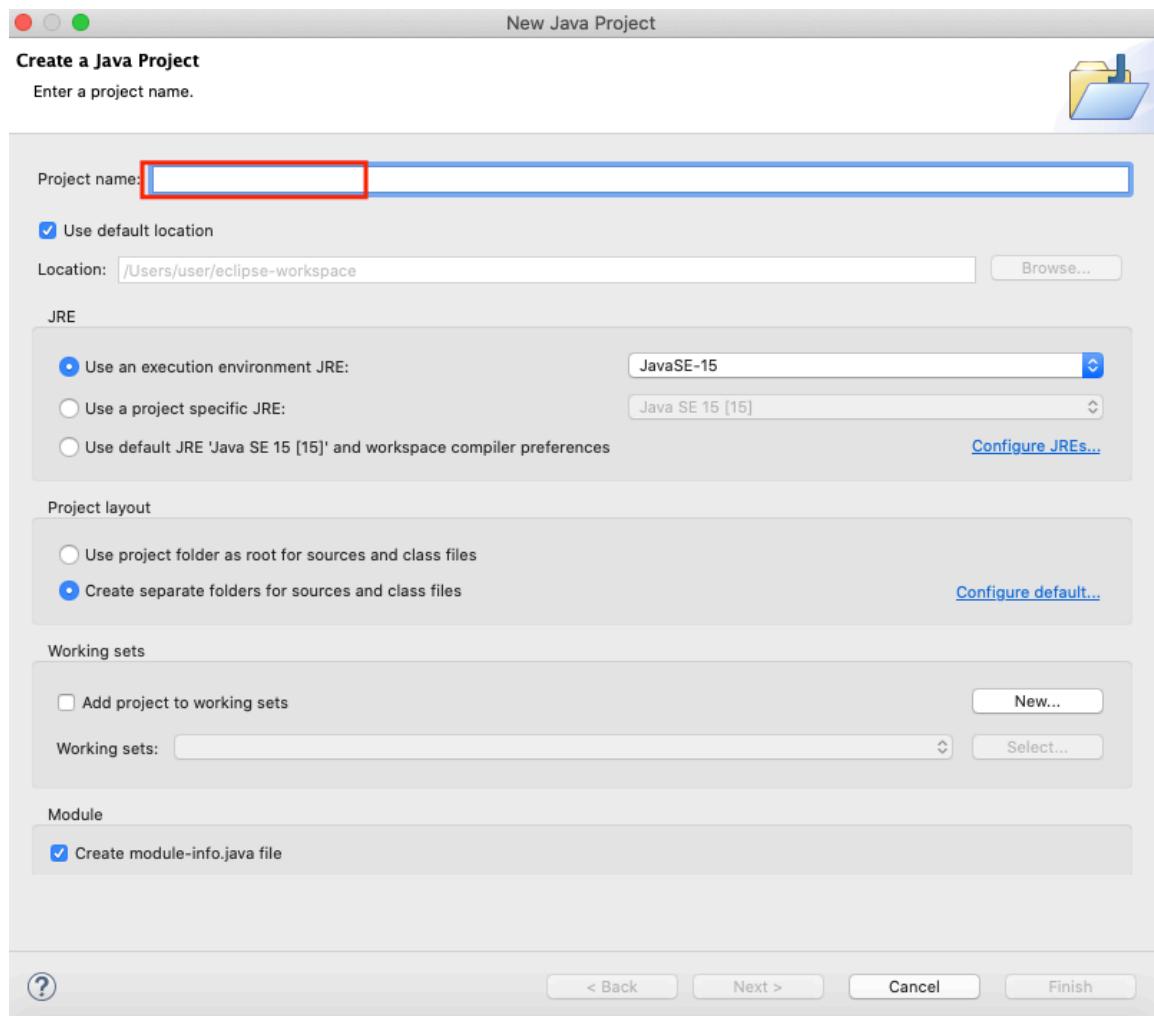
1. Toolbar
2. Package Explorer
3. Workspace
4. Outline

## 5. Console

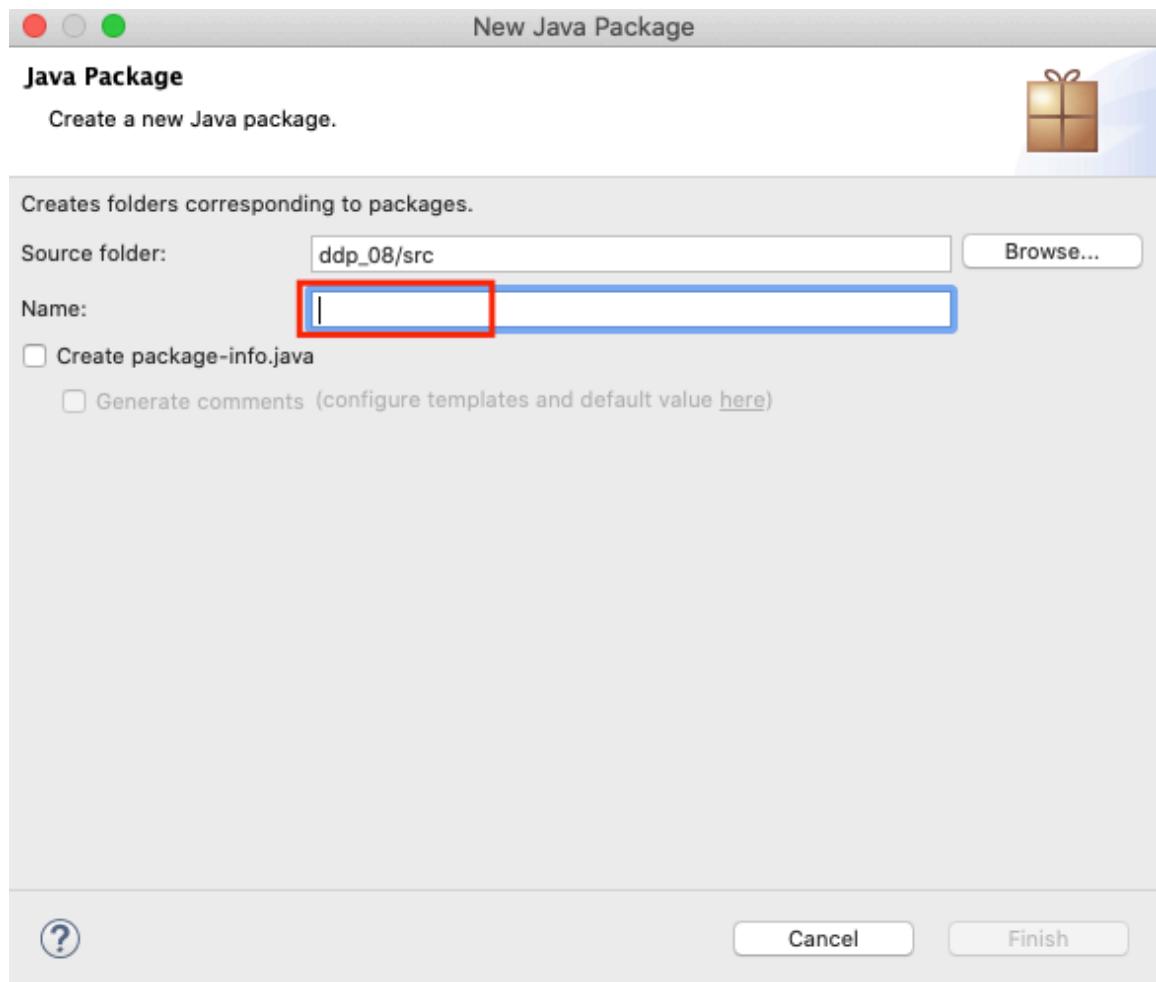
Memulai program dengan eclipase dapat dilakukan dengan membuka aplikasi eclipse terlebih dahulu kemudian jika tampil jendela seperti di bawah ini pilih path tempat menyimpan direktori proyek kemudian pilih launch.



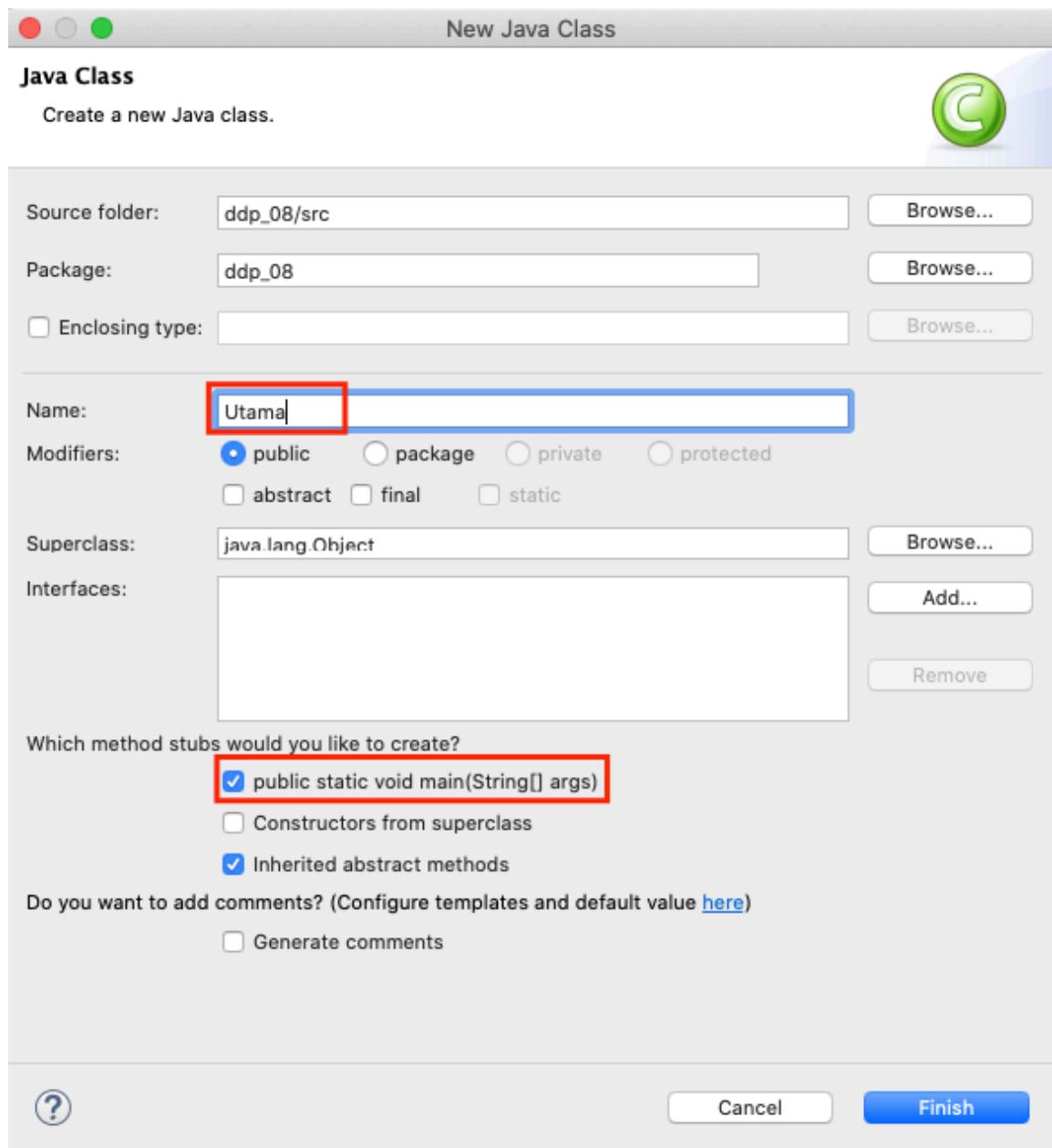
Membuat proyek baru pada eclipse dapat dilakukan dengan memilih file kemudian new pilih java project.



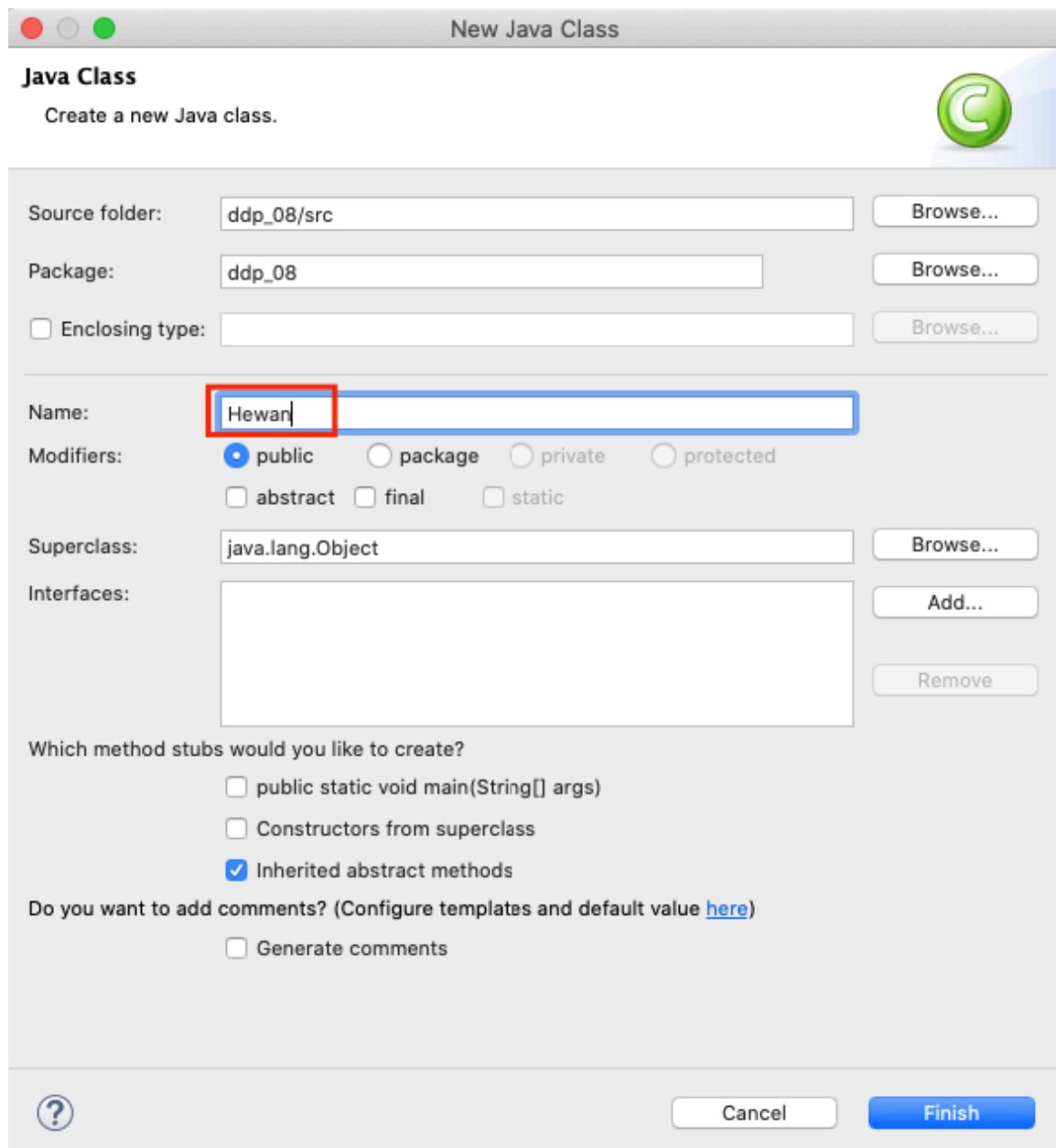
Masukan nama project kemudian finish, selanjutnya adalah membuat package dapat dilakukan dengan klik kanan pada direktori src kemudian pilih new package kemudian tambahkan nama package.



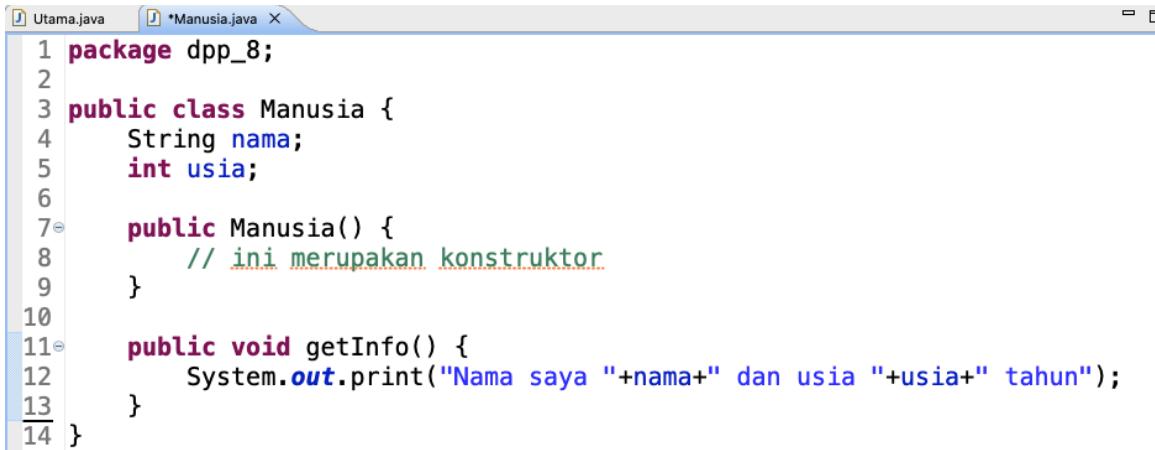
Membuat driver class dapat dilakukan dengan klik kanan pada package kemudian pilih file new class dan sesuaikan dengan form berikut.



Beikutnya adalah membuat object class dapat dilakukan dengan memilih file new class kemudian sesuaikan dengan form berikut ini.

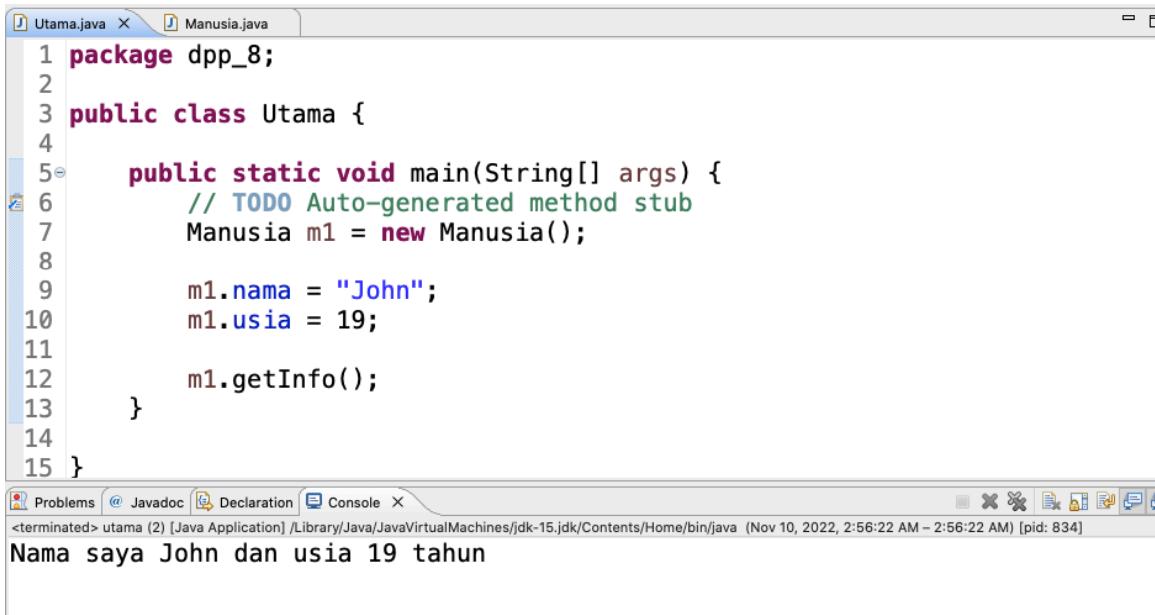


Class object merupakan blueprint atau kerangka yang digunakan sebagai acuan sedangkan driver class nantinya dapat digunakan untuk mengeksekusi object class. Berikut ini merupakan contoh penggunaan object class dan driver class.



```
1 package dpp_8;
2
3 public class Manusia {
4     String nama;
5     int usia;
6
7     public Manusia() {
8         // ini merupakan konstruktor
9     }
10
11    public void getInfo() {
12        System.out.print("Nama saya "+nama+" dan usia "+usia+" tahun");
13    }
14 }
```

Object class ini dapat dieksekusi pada driver class dengan cara seperti berikut ini.



```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Manusia m1 = new Manusia();
8
9         m1.nama = "John";
10        m1.usia = 19;
11
12        m1.getInfo();
13    }
14
15 }
```

Problems Javadoc Declaration Console <terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 2:56:22 AM – 2:56:22 AM) [pid: 834]  
Nama saya John dan usia 19 tahun

Kode pada object class dapat dimodifikasi dengan menambahkan parameter pada konstruktornya seperti berikut ini.

```
1 package dpp_8;
2
3 public class Manusia {
4     String nama;
5     int usia;
6
7     // ini merupakan konstruktor
8     public Manusia(String namanya, int usianya) {
9         nama = namanya;
10        usia = usianya;
11    }
12
13     public void getInfo() {
14         System.out.print("Nama saya "+nama+" dan usia "+usia+" tahun");
15     }
16 }
17 }
```

Kemudian instruksi ini dapat dieksekusi pada driver class dengan menggunakan perintah berikut ini.

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Manusia m1 = new Manusia("John", 19);
8
9         m1.getInfo();
10    }
11
12 }
13
```

Problems Javadoc Declaration Console <terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 3:11:50 AM – 3:11:50 AM) [pid: 908]

Nama saya John dan usia 19 tahun

Pada Java juga terdapat metode accessor dan mutator, metode accessor berarti pengambil (getter) dan metode mutator berarti pengatur (setter). Berikut ini merupakan contoh penggunaannya pada object class.

```
1 package dpp_8;
2
3 public class Manusia {
4     String nama;
5     int usia;
6
7     public void setNama(String n) {
8         this.nama = n;
9     }
10
11    public void setUsia(int u) {
12        this.usia = u;
13    }
14
15    public String getNama() {
16        return this.nama;
17    }
18
19    public int getUsia() {
20        return this.usia;
21    }
22 }
```

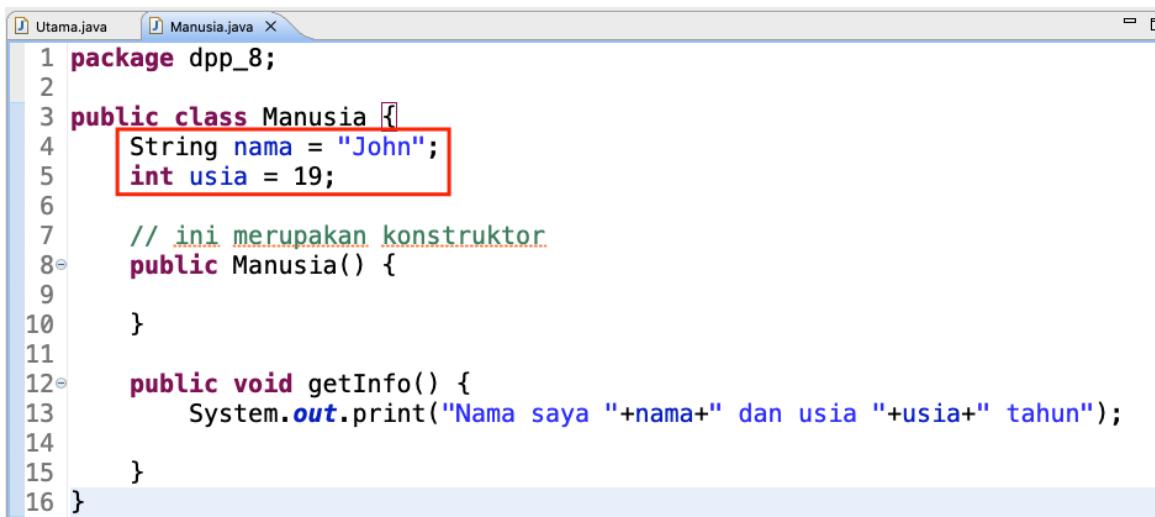
Kemudian untuk mengeksekusinya dapat menggunakan perintah berikut ini yang dituliskan pada driver class.

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Manusia mahasiswa = new Manusia();
8
9         mahasiswa.setNama("Siti");
10        mahasiswa.setUsia(20);
11
12        System.out.println(mahasiswa.getNama());
13        System.out.println(mahasiswa.getUsia());
14
15        Manusia dosen = new Manusia();
16
17        dosen.setNama("Katon");
18        dosen.setUsia(30);
19
20        System.out.println(dosen.getNama());
21        System.out.println(dosen.getUsia());
22    }
23 }
24 }
```

Screenshot of the Eclipse IDE showing the output of the Utama.java program in the Console tab:

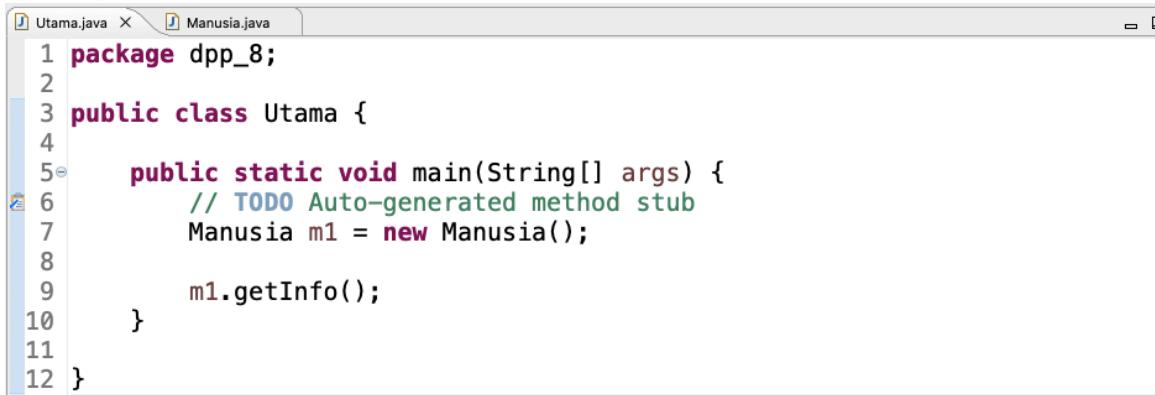
```
<terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 3:05:10 AM – 3:05:10 AM) [pid: 876]
Siti
20
Katon
30
```

Pada dasarnya variabel harus memiliki tipe data primitif seperti string, integer, floating point, double, boolean dan sebagainya. Variabel juga dapat ditambahkan nilai literal seperti berikut ini.



```
1 package dpp_8;
2
3 public class Manusia {
4     String nama = "John";
5     int usia = 19;
6
7     // ini merupakan konstruktor
8     public Manusia() {
9
10    }
11
12     public void getInfo() {
13         System.out.print("Nama saya "+nama+" dan usia "+usia+" tahun");
14     }
15 }
16 }
```

Selanjutnya dapat dieksekusi pada driver class dengan menambahkan perintah berikut ini.



```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Manusia m1 = new Manusia();
8
9         m1.getInfo();
10    }
11
12 }
```

Pada java terdapat casting yang merupakan suatu cara yang digunakan untuk mengubah tipe data primitive. Berikut ini merupakan cara yang digunakan untuk melakukan casting.

The screenshot shows the Eclipse IDE interface with two tabs open: 'Utama.java' and 'Manusia.java'. The 'Utama.java' tab contains the following code:1 package dpp\_8;
2
3 public class Utama {
4
5 public static void main(String[] args) {
6 // TODO Auto-generated method stub
7 double nilai\_uas = 80.2;
8 int uas\_pembulatan = (int)nilai\_uas;
9
10 System.out.println(nilai\_uas);
11 System.out.println(uas\_pembulatan);
12 }
13 }
14
15The output window at the bottom shows the results of the program execution:80.2
80

Pada java juga terdapat casting jenis yang digunakan untuk mengkonversi nilai double ke integer pada bilangan random berikut ini.

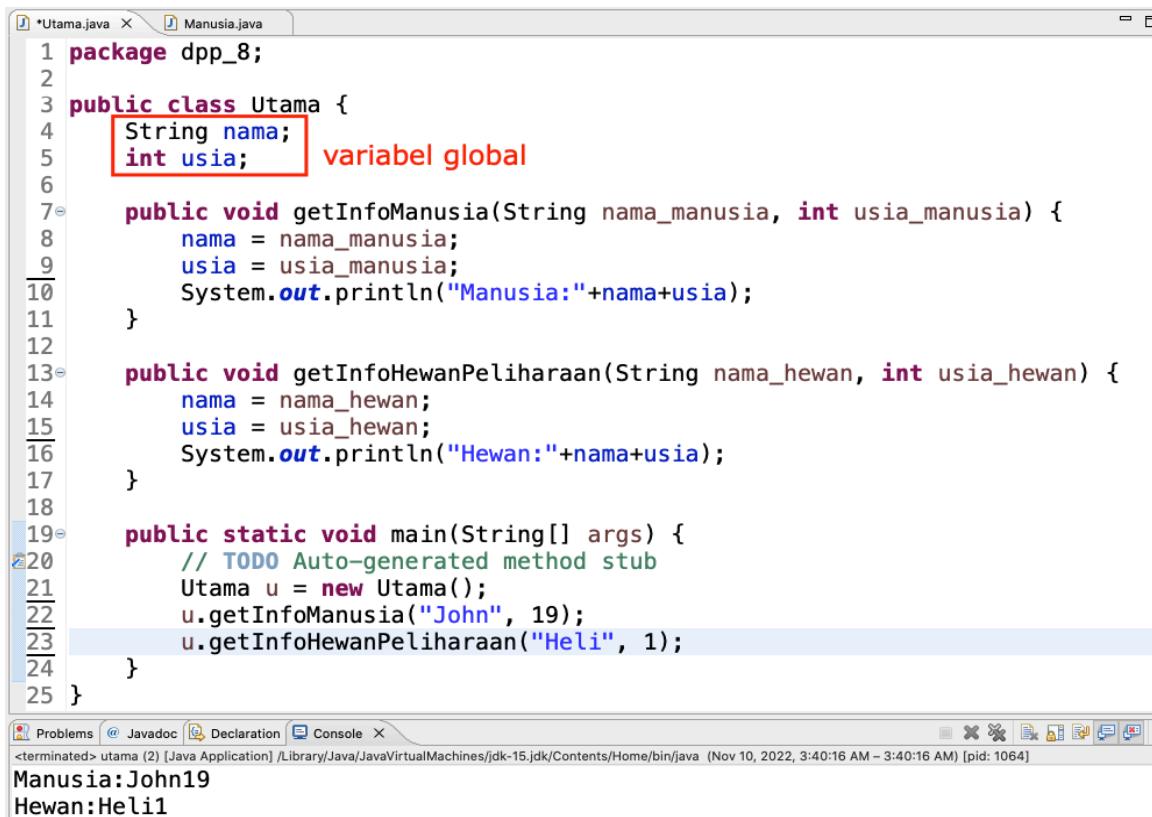
The screenshot shows the Eclipse IDE interface with two tabs open: 'Utama.java' and 'Manusia.java'. The 'Utama.java' tab contains the following code:1 package dpp\_8;
2
3 public class Utama {
4
5 public static void main(String[] args) {
6 int x = (int)(Math.random() \* 10);
7
8 System.out.println(x);
9 }
10 }The output window at the bottom shows the results of the program execution:7

Aturan dalam pembuatan variabel pada java adalah sebagai berikut:

- Tidak boleh menggunakan keword pada java
- Tidak boleh menggunakan spasi
- Tidak boleh menggunakan kombinasi dengan simbol
- Karakter awal tidak boleh berupa angka
- Simbol yang diizinkan hanya tanda underscore (\_) dan tanda dolar (\$)

Selain penulisan variabel, java juga memiliki aturan dalam penulisan konstanta. Konstanta pada java dapat dituliskan dengan menggunakan huruf kapital semua.

Terdapat dua jenis variabel yang ada pada java, yaitu variabel lokal dan variabel global. Variabel lokal hanya dikenali di lingkungan terbatas (sebuah fungsi atau prosedur saja), sedangkan variabel global adalah variabel yang dikenali disemua lingkungan. Berikut ini contoh penerapan variabel lokal dan global.



```
1 package dpp_8;
2
3 public class Utama {
4     String nama;
5     int usia; variabel global
6
7     public void getInfoManusia(String nama_manusia, int usia_manusia) {
8         nama = nama_manusia;
9         usia = usia_manusia;
10        System.out.println("Manusia:"+nama+usia);
11    }
12
13    public void getInfoHewanPeliharaan(String nama_hewan, int usia_hewan) {
14        nama = nama_hewan;
15        usia = usia_hewan;
16        System.out.println("Hewan:"+nama+usia);
17    }
18
19    public static void main(String[] args) {
20        // TODO Auto-generated method stub
21        Utama u = new Utama();
22        u.getInfoManusia("John", 19);
23        u.getInfoHewanPeliharaan("Heli", 1);
24    }
25 }
```

Problems Javadoc Declaration Console <terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 3:40:16 AM – 3:40:16 AM) [pid: 1064]  
Manusia:John19  
Hewan:Heli1

Sedangkan untuk variabel lokal hanya dapat diakses pada metode yang memiliki variabel itu saja yang bisa menggunakannya. Berikut ini penerapan variabel lokal berikut ini contoh implementasinya.

```
1 package dpp_8;
2
3 public class Utama {
4
5     public void getInfoManusia(String nama_manusia, int usia_manusia) {
6         String nama = nama_manusia; → variabel lokal
7         int usia = usia_manusia;
8         System.out.println("Manusia:" + nama+usia);
9     }
10
11    public void getInfoHewanPeliharaan(String nama_hewan, int usia_hewan) {
12 //     nama = nama_hewan; -> variabel nama tidak dapat digunakan
13 //     usia = usia_hewan; -> variabel usia tidak dapat digunakan
14    }
15
16    public static void main(String[] args) {
17        // TODO Auto-generated method stub
18        Utama u = new Utama();
19        u.getInfoManusia("John", 19);
20 //        u.getInfoHewanPeliharaan("Heli", 1);
21    }
22
23 }
```

Problems Javadoc Declaration Console <terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 3:44:28 AM – 3:44:28 AM) [pid: 1083]

Manusia:John19

Java memiliki operator aritmatika yang dapat digunakan untuk mendukung operasi matematika ditambah operasi modulus seperti berikut ini.

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         int x = 8;
7         int y = 4;
8
9         System.out.println(x+y); //penjumlahan
10        System.out.println(x-y); //pengurangan
11        System.out.println(x*y); //perkalian
12        System.out.println(x/y); //pembagian
13        System.out.println(x%y); //modulus
14        System.out.println(Math.pow(x,y)); //Pangkat
15        System.out.println(Math.sqrt(16)); //Akar
16    }
17 }
```

Problems Javadoc Declaration Console <terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 3:57:44 AM – 3:57:44 AM) [pid: 1169]

12  
4  
32  
2  
0  
4096.0  
2.8284271247461903

Operasi increase dan decrease pada java dapat menggunakan simbol plus ganda untuk increase dan minus ganda untuk decrease. Berikut ini contoh penerapannya.

The screenshot shows the Eclipse IDE interface with two tabs open: 'Utama.java' and 'Manusia.java'. The code in 'Utama.java' is as follows:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         int x = 8;
7         int y = 4;
8
9         System.out.println(++x); //Increase
10        System.out.println(--y); //Decrease
11    }
12 }
```

The output window at the bottom shows the results of the console output:

```
9
3
```

Operator penugasan digunakan untuk memasukan nilai ke dalam sebuah variabel. Berikut ini contoh penggunaan operator penugasan.

The screenshot shows the Eclipse IDE interface with two tabs open: 'Utama.java' and 'Manusia.java'. The code in 'Utama.java' is as follows:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         int x = 8;
7         x += 2;
8
9         System.out.println(x);
10    }
11 }
```

The output window at the bottom shows the results of the console output:

```
10
```

Berikutnya adalah operator relasional. Operator ini digunakan untuk membandingkan dua nilai, yaitu sisi kiri dan sisi kanan. Operator ini menggunakan simbol  $<$ ,  $>$ ,  $\leq$ ,  $\geq$ ,  $=$  dan  $\neq$ . Adapun contoh penerapannya pada java adalah sebagai berikut.

The screenshot shows the Eclipse IDE interface. In the top editor window, the code for `Utama.java` is displayed:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         int x = 30;
7         int y = 25;
8
9         System.out.println(x > y);
10        System.out.println(x < y);
11        System.out.println(x == y);
12        System.out.println(x != y);
13    }
14 }
```

In the bottom `Console` tab, the output of the program is shown:

```
true
false
false
true
```

Operator berikutnya adalah operator logika. Operator ini digunakan untuk mengecek logika dari dua kondisi, yaitu kondisi sisi kiri dan kondisi sisi kanan. Operator ini menggunakan simbol `&&` untuk logika and, simbol `||` untuk logika or dan simbol `!` untuk logika not. Berikut ini contoh implementasinya pada java.

The screenshot shows the Eclipse IDE interface. In the top editor window, the code for `Utama.java` is displayed:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         int x = 4;
7         int y = 8;
8         int z = 12;
9
10        System.out.println((x < y) && (y > z));
11        System.out.println((x < y) || (y > z));
12        System.out.println(!(x == y));
13    }
14 }
15 }
```

In the bottom `Console` tab, the output of the program is shown:

```
false
true
true
```

Operator selanjutnya adalah operator ternary, operator ini digunakan untuk pemilihan suatu pernyataan. Jika suatu kondisi bernilai true maka pernyataan pertama akan dieksekusi. Berikut ini contoh penerapannya pada java.

The screenshot shows the Eclipse IDE interface. In the top editor, the code for Utama.java is displayed:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         String status = "active";
7
8         String cek = status == "active" ? "Checked": "None";
9
10        System.out.println(cek);
11    }
12 }
```

In the bottom console tab, the output is shown:

```
Checked
```

Java dapat digunakan untuk melakukan manipulasi string dengan bantuan fungsi built-in yang sudah disediakan seperti fungsi length yang digunakan untuk menghitung panjang string, fungsi substring untuk mengambil karakter dan toUpperCase untuk mengubah ke huruf kapital. Berikut ini contoh implementasinya.

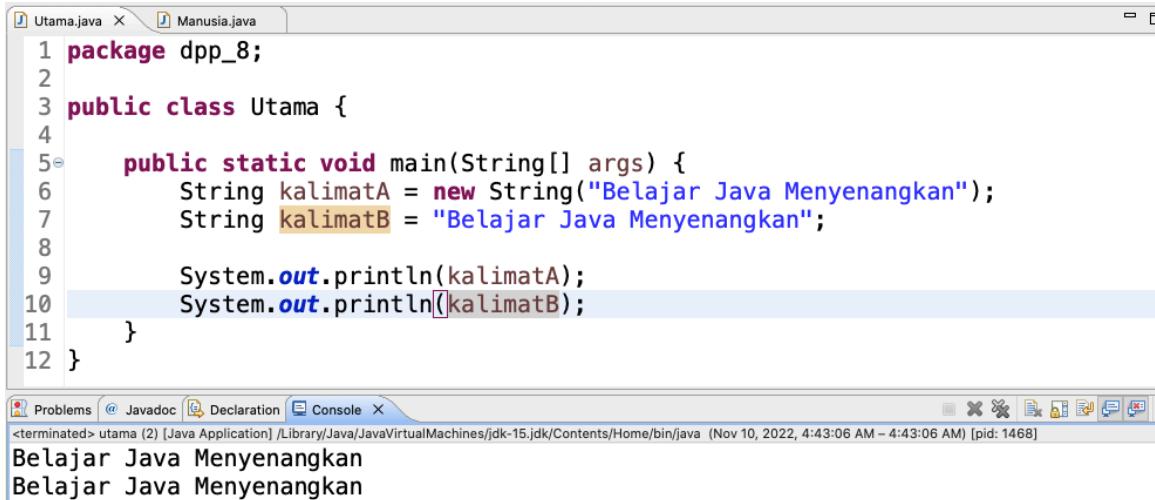
The screenshot shows the Eclipse IDE interface. In the top editor, the code for Utama.java is displayed:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         String kalimat = "Belajar Java Menyenangkan";
7
8         System.out.println(kalimat.length());
9         System.out.println(kalimat.substring(0, 7));
10        System.out.println(kalimat.toUpperCase());
11    }
12 }
```

In the bottom console tab, the output is shown:

```
25
Belajar
BELAJAR JAVA MENYENANGKAN
```

Penulisan string pada java dapat dilakukan dengan menggunakan operator dan juga literal. Berikut ini contoh penerapannya.

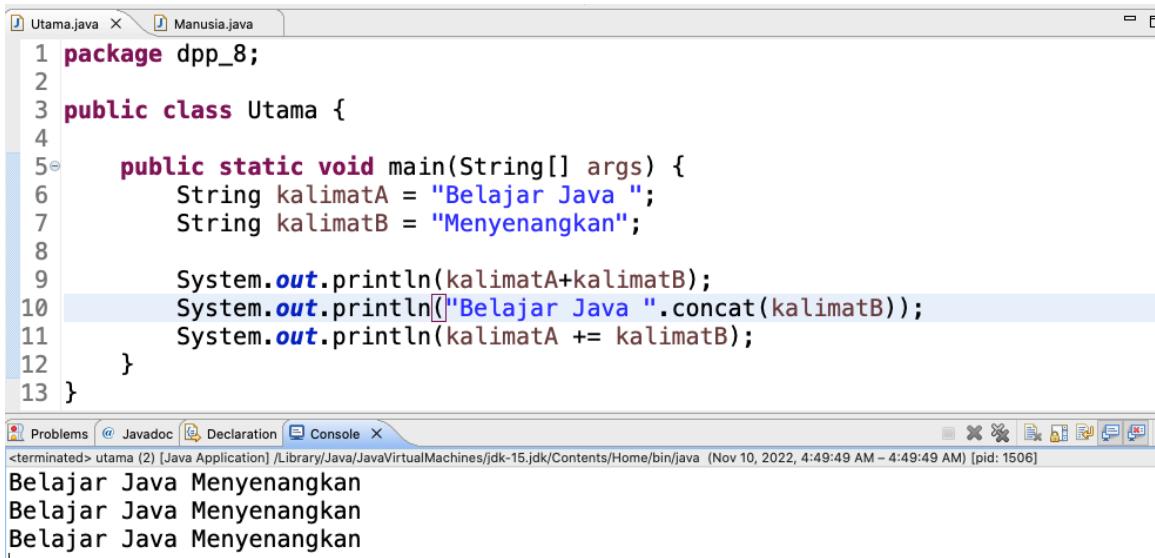


```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         String kalimatA = new String("Belajar Java Menyenangkan");
7         String kalimatB = "Belajar Java Menyenangkan";
8
9         System.out.println(kalimatA);
10        System.out.println(kalimatB);
11    }
12 }
```

Problems Javadoc Declaration Console <terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 4:43:06 AM – 4:43:06 AM) [pid: 1468]

```
Belajar Java Menyenangkan
Belajar Java Menyenangkan
```

Terdapat beberapa metode yang dapat digunakan untuk mengabungkan string, diantaranya adalah dengan menggunakan simbol plus, dengan menggunakan fungsi concat dan yang terakhir adalah dengan menggunakan operator penugasan. Berikut ini merupakan contoh implementasinya.



```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         String kalimatA = "Belajar Java ";
7         String kalimatB = "Menyenangkan";
8
9         System.out.println(kalimatA+kalimatB);
10        System.out.println("Belajar Java ".concat(kalimatB));
11        System.out.println(kalimatA += kalimatB);
12    }
13 }
```

Problems Javadoc Declaration Console <terminated> utama (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 10, 2022, 4:49:49 AM – 4:49:49 AM) [pid: 1506]

```
Belajar Java Menyenangkan
Belajar Java Menyenangkan
Belajar Java Menyenangkan
```

Java menyediakan metode yang dapat digunakan untuk membandingkan dua string. Berikut ini contoh penerapannya pada java. Metode yang pertama adalah compareTo() digunakan untuk menentukan urutan leksografis dua string, equals() digunakan untuk mengecek kesamaan dua string. Berikut contoh penerapannya.

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         String kataA = "ABCD";
7         String kataB = "ABC";
8
9         System.out.println(kataA.compareTo(kataB));
10        System.out.println(kataA.equals(kataB));
11    }
12 }
```

1  
false

Mencari karakter dapat dilakukan dengan menggunakan bantuan fungsi indexOf, fungsi ini dapat mengecek keberadaan suatu karakter berdasarkan index-nya. Berikut implementasinya pada java.

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         String karakter = "ABCDEFGHI";
7
8         String cari = "E";
9
10        System.out.println("Posisi karakter "+cari+" berada pada urutan ke "+
11                            +(karakter.indexOf(cari)+1));
12 }
```

Posisi karakter E berada pada urutan ke 5

Java mendukung masukan yang di kirim dari keyboard, namun untuk membaca masukan yang di kirim melalui keyboard harus menggunakan pustaka scanner. Adapun cara penggunaan pustaka scanner seperti berikut ini.

```

1 package dpp_8;
2 import java.util.Scanner;
3
4 public class Utama {
5
6     public static void main(String[] args) {
7         String nama;
8         int usia;
9
10        Scanner s = new Scanner(System.in);
11
12        System.out.print("Masukan Nama: ");
13        nama = s.nextLine();
14
15        System.out.print("Masukan Usia: ");
16        usia = s.nextInt();
17
18        System.out.println("Nama Anda: "+nama);
19        System.out.println("Usia Anda: "+usia);
20
21        s.close();
22    }
23 }

```

The screenshot shows the Eclipse IDE interface with two tabs open: "Utama.java" and "Manusia.java". The "Utama.java" tab is active, displaying the provided Java code. Below the code editor is the "Console" view, which shows the output of running the program. The console output reads:

```

Masukan Nama: Siti
Masukan Usia: 17
Nama Anda: Siti
Usia Anda: 17

```

Pernyataan if merupakan blok pernyataan untuk pemilihan, jika suatu kondisi bernilai true maka pernyataannya akan dieksekusi. Ada beberapa contoh penggunaan pernyataan if seperti berikut ini.

If -> terdapat satu kondisi dan satu pernyataan

```

1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         int bilangan = 8;
7
8         if (bilangan > 0) {
9             System.out.println("Bilangan Positif");
10        }
11    }
12 }

```

The screenshot shows the Eclipse IDE interface with the same Java code as the previous example. The "Console" view shows the output of running the program, which is:

```

Bilangan Positif

```

If-else if -> terdapat satu kondisi dan dua pernyataan

The screenshot shows the Eclipse IDE interface with two tabs open: 'Utama.java' and 'Manusia.java'. The code in 'Utama.java' is:

```
3 public class Utama {  
4  
5     public static void main(String[] args) {  
6         int bilangan = -3;  
7  
8         if (bilangan > 0) {  
9             System.out.println("Bilangan Positif");  
10        } else {  
11            System.out.println("Bilangan Negatif");  
12        }  
13    }  
14 }
```

The output in the 'Console' tab is: **Bilangan Negatif**.

If-else if-else -> terdapat beberapa kondisi dan beberapa pernyataan

The screenshot shows the Eclipse IDE interface with two tabs open: '\*Utama.java' and 'Manusia.java'. The code in '\*Utama.java' is:

```
1 package dpp_8;  
2  
3 public class Utama {  
4  
5     public static void main(String[] args) {  
6         int bilangan = -1;  
7  
8         if (bilangan > 0) {  
9             System.out.println("Bilangan Positif");  
10        } else if (bilangan < 0) {  
11            System.out.println("Bilangan Negatif");  
12        } else {  
13            System.out.println("Bilangan Nol");  
14        }  
15    }  
16 }
```

The output in the 'Console' tab is: **Bilangan Negatif**.

Nested if -> terdapat if di dalam suatu pernyataan if

The screenshot shows the Eclipse IDE interface with two tabs open: 'Utama.java' and 'Manusia.java'. The code in 'Utama.java' is as follows:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         String username_db = "jhon";
7         String password_db = "jhon123";
8
9         String username = "jhon";
10        String password = "jhon123";
11
12        if (username == username_db) {
13            if (password == password_db) {
14                System.out.println("Berhasil Login");
15            } else {
16                System.out.println("Password Salah");
17            }
18        } else {
19            System.out.println("Username Salah");
20        }
21    }
22 }
```

The output in the 'Console' tab is: Berhasil Login

Selain pernyataan if, java juga memiliki switch case yang dapat digunakan untuk memilih, cara kerja switch case berbeda dengan pernyataan if, jika pernyataan if mengecek kondisi satu per satu, maka switch case langsung mengeksekusi apa yang menjadi pilihan. Switch case lebih tepat digunakan untuk navigasi menu. Berikut ini contoh penerapannya pada java.

The screenshot shows the Eclipse IDE interface with two tabs open: 'Utama.java' and 'Manusia.java'. The code in 'Utama.java' is as follows:

```
1 package dpp_8;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6         int menu = 3;
7
8         switch(menu) {
9             case 1:
10                 System.out.println("Anda Memilih Menu 1");
11                 break;
12             case 2:
13                 System.out.println("Anda Memilih Menu 2");
14                 break;
15             case 3:
16                 System.out.println("Anda Memilih Menu 3");
17                 break;
18             default:
19                 System.out.println("Menu Tidak Ada");
20             }
21     }
22 }
```

The output in the 'Console' tab is: Anda Memilih Menu 3

Penggunaan switch case juga bisa dikombinasi dengan struktur lainnya seperti if – else. Berikut ini contoh penerapannya.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         int menu = 1;
8
9         switch(menu) {
10             case 1:
11                 int tahun = 4;
12                 if ((tahun > 1) && (tahun < 3.5)) {
13                     System.out.println("Cumlaude");
14                 } else {
15                     System.out.println("Bukan Cumlaude");
16                 }
17                 break;
18             case 2:
19                 double nilai = 2.5;
20                 if ((nilai > 3.51) && (nilai < 4)) {
21                     System.out.println("Cumlaude");
22                 } else {
23                     System.out.println("Bukan Cumlaude");
24                 }
25                 break;
26             default:
27                 System.out.println("Menu Tidak Ada");
28         }
29     }
30 }
```

Selain pernyataan kontrol dalam bentuk percabangan, dalam pemrograman java juga terdapat pernyataan kontrol dalam bentuk perulangan. Perulangan dapat membantu untuk mengeksekusi suatu tindakan secara berulang-ulang. Dalam java terdapat tiga bentuk perulangan yang digunakan, diantaranya adalah perulangan for, while dan do while. Perulangan for dapat dibuat dalam dua bentuk, yaitu bentuk increment dan decrement. Berikut ini merupakan contoh penggunaan perulangan for pada java.

The screenshot shows the Eclipse IDE interface. The top part displays the code for `Utama.java`. The code contains two `for` loops: one for incrementing `i` from 1 to 5, and another for decrementing `i` from 5 to 1. The bottom part shows the `Console` view with the output of the program, which is the numbers 1 through 5 printed on separate lines.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         // INCREMENT
8         for(int i=1;i<5;i++) {
9             System.out.println(i);
10        }
11
12     // DECREMENT
13     for(int i=5;i>1;i--) {
14         System.out.println(i);
15     }
16 }
17 }
```

```
1
2
3
4
5
4
3
2
```

Perulangan lainnya adalah perulangan while, perulangan memiliki format yang berbeda dengan bentuk perulangan for. Perulangan ini juga dapat dibuat dalam dua bentuk seperti pada perulangan for sebelumnya, adapun bentuk dari perulangan ini adalah increment dan decrement. Berikut ini merupakan contoh penerapan perulangan for pada java.

```
*Utama.java X Manusia.java
1 package ddp_09;
2 public class Utama {
3
4     public static void main(String[] args) {
5
6         // INCREMENT
7         int i = 1;
8         while(i<5) {
9             System.out.println(i);
10            i += 1;
11        }
12
13        // DECREMENT
14        int j = 5;
15        while(j>1) {
16            System.out.println(j);
17            j -= 1;
18        }
19    }
20 }

Problems Javadoc Declaration Console <terminated> utama (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 25, 2022, 10:35:31 PM – 10:35:31 PM) [pid: 1]
1
2
3
4
5
4
3
2
```

Perulangan yang terakhir yang dapat digunakan pada java adalah perulangan do while. Perulangan ini merupakan bentuk lain dari perulangan while. Adapun contoh penggunaannya seperti yang tampak pada gambar berikut ini.

```
1 package ddp_09;
2 public class Utama {
3
4     public static void main(String[] args) {
5         // INCREMENT
6         int i = 1;
7         do {
8             System.out.println(i);
9             i += 1;
10        } while(i < 5);
11
12        // DECREMENT
13        int j = 5;
14        do {
15            System.out.println(j);
16            j -= 1;
17        } while(j > 1);
18    }
19 }
```

Problems @ Javadoc Declaration Console

<terminated> utama (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-15.jdk/Contents/Home/bin/java (Nov 25, 2022, 10:41:10 PM – 10:41:11 PM) [pid: 14]

```
1
2
3
4
5
4
3
2
```

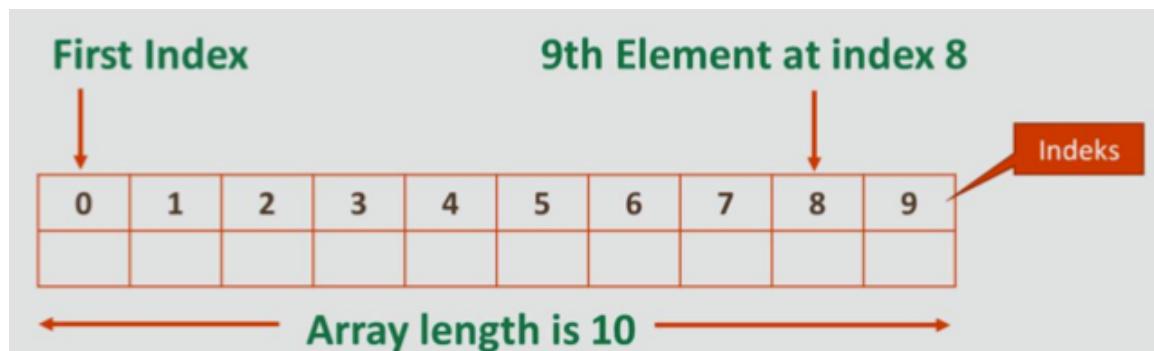
Dalam pemrograman java terdapat *control statement* yang dapat digunakan untuk mengendalikan sebuah *statement*. Terdapat dua control statement yang dapat digunakan, yaitu break dan continue. Perhatikan contoh penerapan penyataan break berikut ini.

```
1 package ddp_09;
2 public class Utama {
3
4     public static void main(String[] args) {
5
6         for(int i=1;i<10;i++) {
7             if (i % 2 == 0) {
8                 System.out.println(i);
9                 break;
10            }
11        }
12    }
13 }
```

Pernyataan break akan menghentikan suatu pernyataan jika suatu kondisi terpenuhi atau jika kondisi bernilai true, sedangkan continue melanjutkan ke pernyataan berikutnya. Berikut ini contoh penggunaan pernyataan continue.

```
1 package ddp_09;
2 public class Utama {
3
4     public static void main(String[] args) {
5
6         for(int i=1;i<10;i++) {
7             if (i % 2 == 0) {
8                 System.out.println(i);
9                 continue;
10            }
11        }
12    }
13 }
```

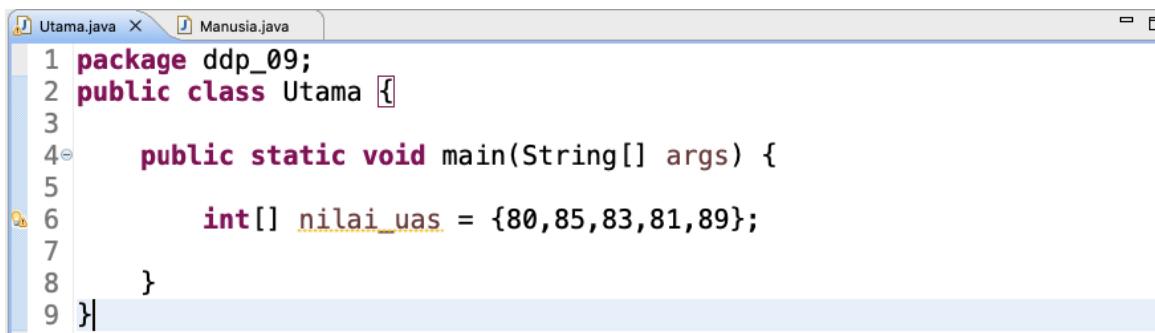
Dalam pemrograman terdapat teknik yang dapat digunakan untuk menampung beberapa nilai dalam sebuah variabel, variabel ini disebut sebagai variabel array. Berikut ini merupakan penampakan dari array.



Dalam array terdapat elemen dan juga index, elemen menyatakan suatu nilai yang disimpan, sedangkan index merupakan urutan dari elemen tersebut dan dimulai dari angka nol. Java mendukung dalam pembuatan array seperti berikut ini.

```
1 package ddp_09;
2 public class Utama {
3
4     public static void main(String[] args) {
5
6         int[] nilai_uas = new int[5];
7
8         nilai_uas[0] = 80;
9         nilai_uas[1] = 85;
10        nilai_uas[2] = 83;
11        nilai_uas[3] = 81;
12        nilai_uas[4] = 89;
13    }
14 }
15 }
```

Selain menggunakan teknik seperti di atas dalam membuat array, terdapat teknik lain yang dapat digunakan seperti di bawah ini.

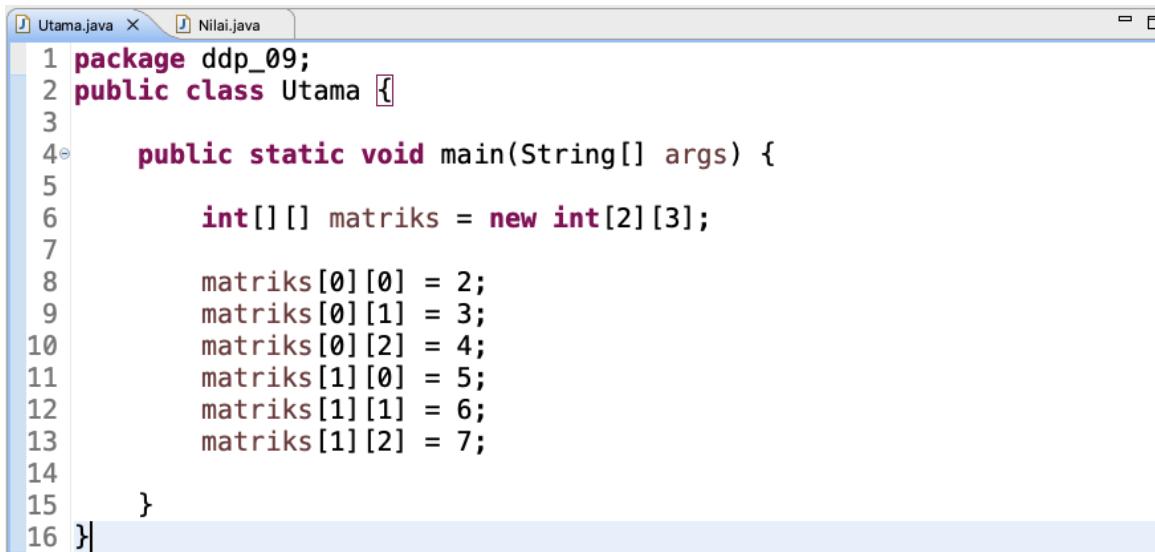


```
Utama.java X Manusia.java
1 package ddp_09;
2 public class Utama {
3
4     public static void main(String[] args) {
5
6         int[] nilai_uas = {80,85,83,81,89};
7
8     }
9 }
```

Untuk dapat mengakses array ini dapat menggunakan index jika hanya ingin mendapatkan satu nilai tertentu saja dan jika ingin mengakses semua data dapat menggunakan perulangan.

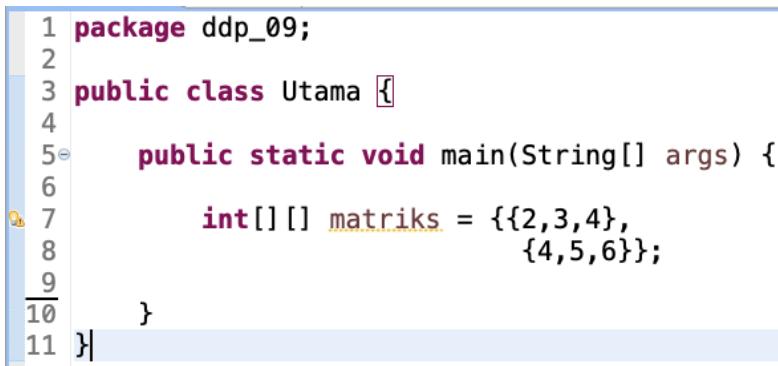
```
15     System.out.println(nilai[3]);
16
17     for(int i = 0;i < 5;i++) {
18         System.out.println(nilai[i]);
19     }
```

Selain array satu dimensi, java juga mendukung pembuatan array dua dimensi. Dua dimensi pada array artinya terdapat baris dan kolom. Berikut ini contoh penerapan array dua dimensi pada java.



```
1 package ddp_09;
2 public class Utama {
3
4     public static void main(String[] args) {
5
6         int[][] matriks = new int[2][3];
7
8         matriks[0][0] = 2;
9         matriks[0][1] = 3;
10        matriks[0][2] = 4;
11        matriks[1][0] = 5;
12        matriks[1][1] = 6;
13        matriks[1][2] = 7;
14    }
15 }
16 }
```

Adapun cara lain yang dapat digunakan untuk membuat array dua dimensi adalah dengan langsung melakukan inisialisasi nilai, tanpa perlu menentukan jumlah baris dan kolom array.



```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         int[][] matriks = {{2,3,4},
8                             {4,5,6}};
9
10    }
11 }
```

Untuk mengakses array dua dimensi dengan nilai tertentu dapat menggunakan teknik yang sama dengan cara mengakses array satu dimensi, hanya saja pada array ada penambahan baris sehingga terdapat dua kurung siku, sedangkan untuk mengakses semua data dalam array dua dimensi dapat menggunakan nested looping atau perulangan bersarang, dimana dalam perulangan terdapat perulangan lagi.

```

16     System.out.println(nilai[1][0]);
17
18     for(int i = 0;i < nilai.length;i++) {
19         for(int j = 0;j< nilai[i].length;j++) {
20             System.out.print(nilai[i][j]);
21         }
22         System.out.println();
23     }

```

Dalam menangani kelasahan yang mungkin saja terjadi dalam pemrograman, java memiliki teknik penanganan galat. Galat mengindikasikan bahwa terdapat kesalahan pada penerapan program. Terdapat tiga jenis kesalahan, diantaranya adalah kesalahan sintaks, kesalahan logika dan kesalahan *run-time*. Kesalahan sintaks dan kesalahan logika ini berarti kesalahan yang bisa saja dilakukan oleh pengembang, sedangkan kesalahan run-time merupakan kesahan yang terjadi karena keterbatasan program, namun keasalahan run-time ini dapat diatasi dengan menggunakan try-exception.

```

1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         try {
8             int x = 4;
9             int y = 0;
10
11             System.out.println(x/y);
12         } catch(Exception e) {
13             System.out.println("Terjadi Kesalahan Run-Time");
14             System.out.println("Kesalahan Terdeteksi: "+e);
15         }
16     }
17 }
18

```

Kesalahan operasi aritmatika seperti ini juga dapat dibuat lebih spesifik dengan penanganan galat seperti berikut.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         try {
8             System.out.println(2/0);
9         } catch (ArithmaticException e) {
10            System.out.println("Pembagian Nol");
11        } finally {
12            System.out.println("Jalankan Operasi Wajib");
13        }
14    }
15 }
16 }
```

Java juga menyediakan pengecualian untuk masukan dan juga keluaran, ini dapat digunakan untuk mengecek keberadaan suatu file seperti berikut ini.

```
1 package ddp_09;
2 import java.io.FileReader;
3 import java.io.IOException;
4
5 public class Utama {
6
7     public static void main(String[] args) {
8
9         try {
10            FileReader read = new FileReader("file.txt");
11        } catch (IOException e) {
12            System.out.println("File Tidak Ditemukan");
13            System.out.println(e);
14        }
15    }
16 }
17 }
```

Pengembang juga bisa mengatasi galat dengan menggunakan metode throws exception yang sudah disediakan seperti berikut ini.

```
1 package ddp_09;
2 import java.io.FileReader;
3 import java.io.IOException;
4
5 public class Utama {
6
7     public static void main(String[] args) throws IOException {
8
9         FileReader read = new FileReader("file.txt");
10
11    }
12 }
```

Pada penanganan galat terdapat kata kunci finally yang digunakan pada akhir blok, dimana pernyataan pada blok ini selalu dijalankan dan tidak bergantung pada pernyataan pada blok pertama.

```
1 package ddp_09;
2 import java.io.FileReader;
3 import java.io.IOException;
4
5 public class Utama {
6
7     public static void main(String[] args) {
8
9         try {
10             FileReader read = new FileReader("file.txt");
11         } catch (IOException e) {
12             System.out.println("File Tidak Ditemukan");
13         } finally {
14             System.out.println("Jalankan Operasi Wajib");
15         }
16     }
17 }
18 }
```

Java merupakan bahasa pemrograman yang menggunakan paradigma pemrograman berorientasi objek, sehingga pada Java dapat dibuat class, object dan metode. Adapun contoh pembuatan class dan metode pada Java seperti berikut ini.

```
1 package ddp_09;
2
3 public class Manusia {
4     String nama;
5     int usia;
6
7     public Manusia(String n, int u) {
8         this.nama = n;
9         this.usia = u;
10    }
11
12    public void identitas() {
13        System.out.println("Nama "+nama+" dan usia "+usia);
14    }
15
16    public int tahunLahir() {
17        return 2022-17;
18    }
19 }
```

Metode dalam class Manusia tersebut hanya bisa digunakan jika sudah menambahkan objek seperti berikut ini.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Manusia m = new Manusia("Siti", 17);
8
9         m.identitas();
10
11        System.out.println("Siti lahir tahun "+m.tahunLahir());
12
13    }
14 }
```

Membuat metode mutator dan accessor secara otomatis pada java dapat dilakukan dengan menambahkan property atau variabel global terlebih dahulu, kemudian pilih menu source dan pilih generate getter and setter, centang semua property yang dikehendaki kemudian klik generate.

```
1 package ddp_09;
2
3 public class Manusia {
4     String nama;
5     int usia;
6     public String getNama() {
7         return nama;
8     }
9     public void setNama(String nama) {
10        this.nama = nama;
11    }
12     public int getUsia() {
13         return usia;
14     }
15     public void setUsia(int usia) {
16         this.usia = usia;
17     }
```

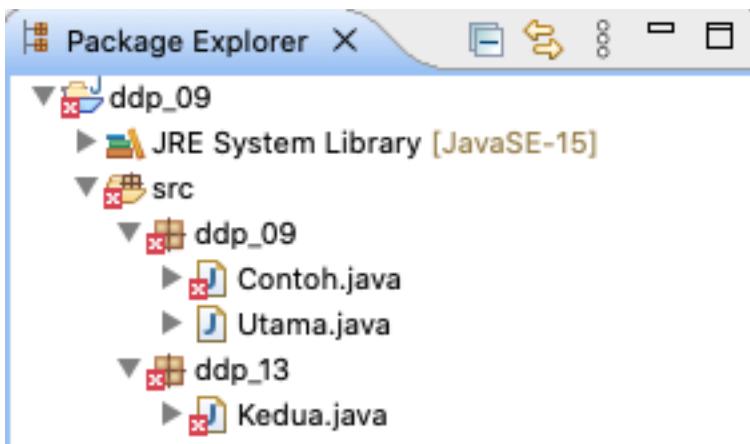
Adapun cara menggunakan metode mutator dan accessor adalah dengan menambahkan objeknya terlebih dahulu, kemudian isi nilai pada metode setter dan tampilkan nilainya dengan getter seperti berikut ini.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Manusia m = new Manusia();
8
9         m.setNama("Siti");
10        m.setUsia(17);
11
12        System.out.println(m.getNama());
13        System.out.println(m.getUsia());
14
15    }
16 }
```

Pada Java terdapat access modifier atau pengubah akses, diantaranya adalah seperti berikut ini.

Pengubah Akses	Description
public	Mengizinkan akses dari mana saja.
protected	Mengizinkan akses hanya dari dalam kelas yang sama, dari subclass, atau dari kelas lain paket yang sama dengan pengubah.
private	Mengizinkan akses hanya di dalam kelas yang sama dengan pengubah.
"default" (tidak ditetapkan/kosong)	Mengizinkan akses hanya di dalam kelas yang sama, atau dari kelas lain paket yang sama dengan pengubah.

Untuk membuktikan hal tersebut, perlu dilakukan pengujian dengan menggunakan java seperti berikut ini.



Buatlah dua package dan juga tambahkan beberapa class di dalamnya untuk membuktikan access modifier pada package yang berbeda. Selanjutnya tambahkan beberapa metode pengujian untuk masing-masing class seperti berikut ini.

```
3 public class Utama {  
4  
5     public void contohPublic() {  
6         System.out.println("Contoh Public");  
7     }  
8  
9     private void contohPrivate() {  
10        System.out.println("Contoh Private");  
11    }  
12  
13     protected void contohProtected() {  
14        System.out.println("Contoh Protected");  
15    }  
16  
17     void contohDefault() {  
18        System.out.println("Contoh Default");  
19    }  
20  
21     public static void main(String[] args) {  
22  
23         Utama u = new Utama();  
24  
25         u.contohPublic();  
26         u.contohPrivate();  
27         u.contohProtected();  
28         u.contohDefault();
```

Ini menunjukkan semua access modifier dapat digunakan karena berada pada class yang sama dan juga package yang sama, bagaimana jika berada pada class yang berbeda seperti berikut ini.

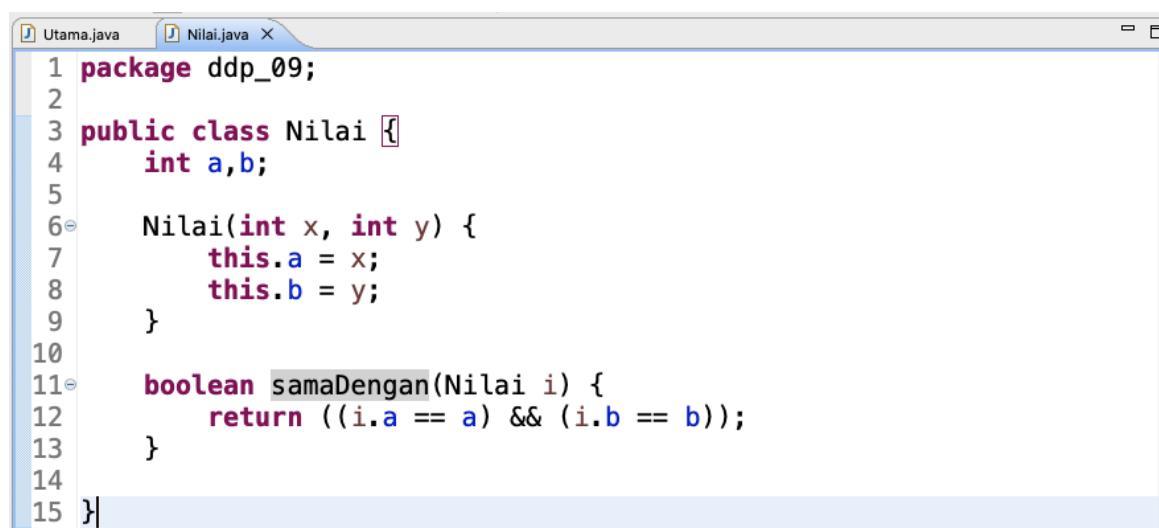
```
1 package ddp_09;  
2  
3 public class Contoh {  
4     public static void main(String[] args) {  
5  
6         Utama u = new Utama();  
7  
8         u.contohPublic();  
9         u.contohPrivate(); // Error: cannot find symbol  
10        u.contohProtected();  
11        u.contohDefault();  
12    }  
13 }  
14 |
```

Ini menunjukkan bahwa access modifier private tidak bisa digunakan pada class yang berbeda walaupun masih dalam satu package, bagaimana dengan class yang berbeda dan package yang berbeda seperti berikut ini.

```
1 package ddp_13;
2 import ddp_09.Utama;
3
4 public class Kedua {
5
6     public static void main(String[] args) {
7
8         Utama u = new Utama();
9
10        u.contohPublic();
11        u.contohPrivate();
12        u.contohProtected();
13        u.contohDefault();
14    }
15
16
17 }
```

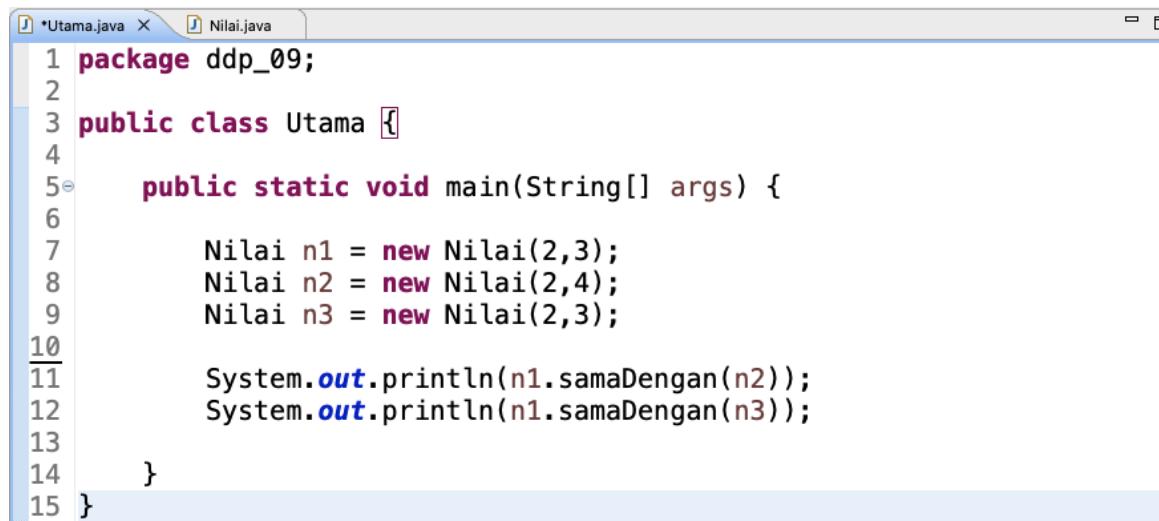
Pada kasus ini access modifier private, protected dan default tidak bisa digunakan karena berada pada class lain pada package yang berbeda.

Pada java, tipe parameter dapat berupa sebuah objek seperti halnya menjadikan objek sebagai tipe variabel.



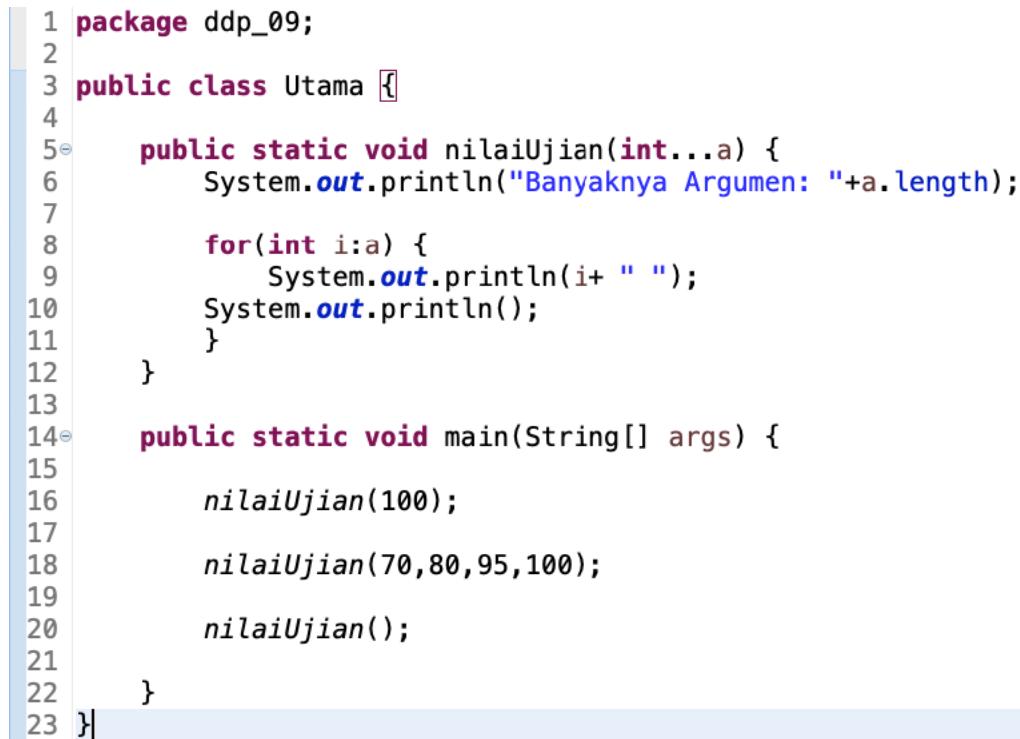
```
Utama.java Nilai.java
1 package ddp_09;
2
3 public class Nilai {
4     int a,b;
5
6     Nilai(int x, int y) {
7         this.a = x;
8         this.b = y;
9     }
10
11    boolean samaDengan(Nilai i) {
12        return ((i.a == a) && (i.b == b));
13    }
14
15 }
```

Pada kasus tersebut, i bernilai objek sehingga jika objek dari metode sama dengan dibuat akan menghasilkan sebuah objek. Berikut ini cara penggunaannya.



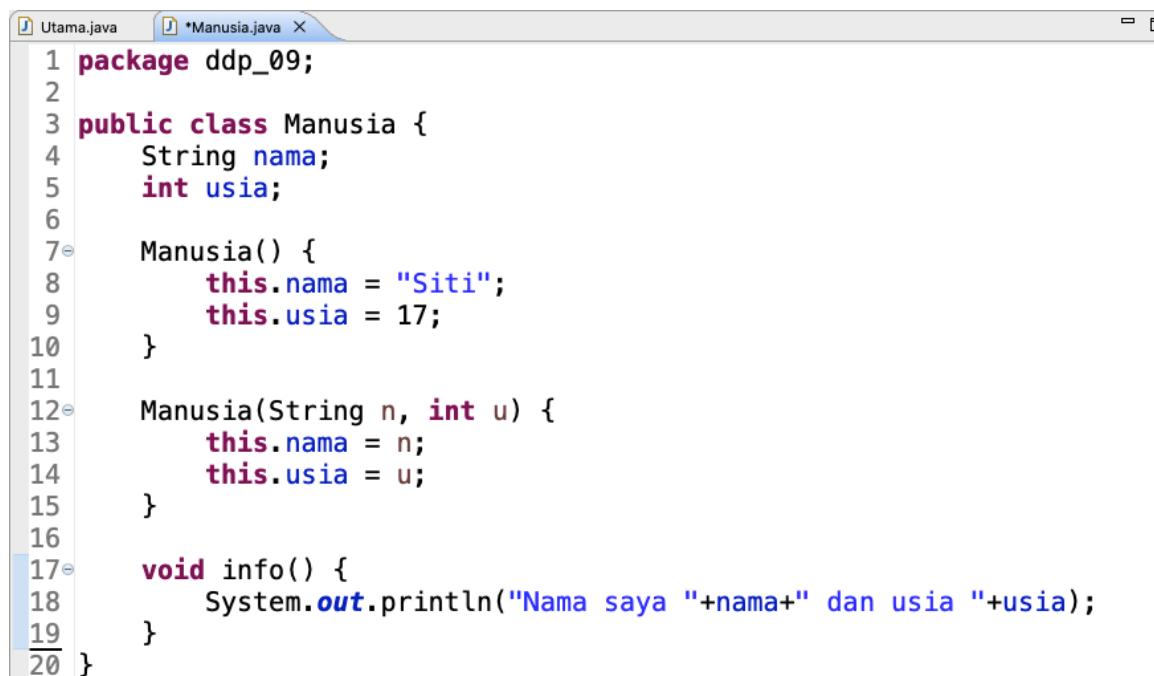
```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Nilai n1 = new Nilai(2,3);
8         Nilai n2 = new Nilai(2,4);
9         Nilai n3 = new Nilai(2,3);
10
11        System.out.println(n1.samaDengan(n2));
12        System.out.println(n1.samaDengan(n3));
13    }
14 }
15 }
```

Pada java juga mendukung pembuatan variabel argumen atau varags. Berikut ini contoh penggunaannya.



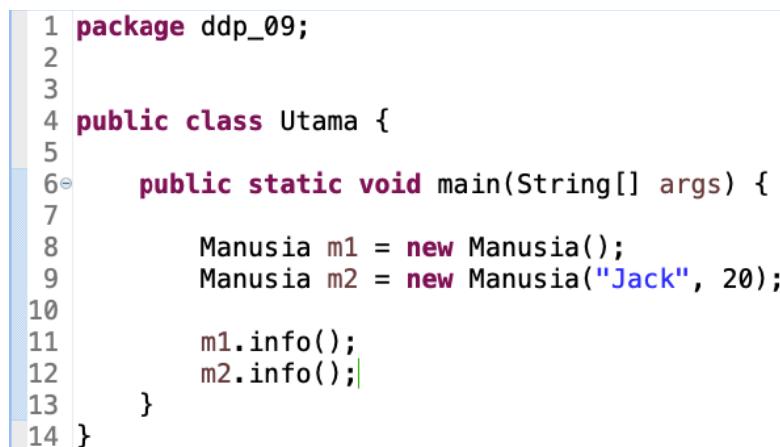
```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void nilaiUjian(int...a) {
6         System.out.println("Banyaknya Argumen: "+a.length);
7
8         for(int i:a) {
9             System.out.println(i+ " ");
10            System.out.println();
11        }
12    }
13
14    public static void main(String[] args) {
15
16        nilaiUjian(100);
17
18        nilaiUjian(70,80,95,100);
19
20        nilaiUjian();
21
22    }
23 }
```

Overload pada java dapat diartikan sebagai beberapa metode dengan nama yang sama dalam satu class. Berikut ini contoh konstruktor pada metode overload.



```
1 package ddp_09;
2
3 public class Manusia {
4     String nama;
5     int usia;
6
7     Manusia() {
8         this.nama = "Siti";
9         this.usia = 17;
10    }
11
12    Manusia(String n, int u) {
13        this.nama = n;
14        this.usia = u;
15    }
16
17    void info() {
18        System.out.println("Nama saya "+nama+" dan usia "+usia);
19    }
20 }
```

Pada class Manusia terdapat dua metode konstruktor dengan nama yang sama, ini disebut sebagai konstruktor overload. Overload tidak hanya bisa digunakan pada metode overload, namun juga bisa digunakan pada metode yang lain. Adapun cara menggunakannya adalah dengan membuat objek seperti di bawah ini.



```
1 package ddp_09;
2
3
4 public class Utama {
5
6     public static void main(String[] args) {
7
8         Manusia m1 = new Manusia();
9         Manusia m2 = new Manusia("Jack", 20);
10
11         m1.info();
12         m2.info();
13     }
14 }
```

Contoh lain penggunaan overload pada metode selain metode konstruktor adalah sebagai berikut.

```
1 package ddp_09;
2
3 public class Manusia {
4
5     void objek() {
6         System.out.println("Objek manusia memiliki nama");
7     }
8
9     void objek(String nama) {
10        System.out.println("Contoh nama objek manusia "+nama);
11    }
12 }
```

Adapun penggunaannya dapat menggunakan satu buah objek seperti berikut ini.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Manusia m = new Manusia();
8
9         m.objek();
10        m.objek("Siti");
11    }
12 }
```

Dalam pemrograman java, kata kunci static digunakan untuk mengakses variabel atau metode pada class tertentu tanpa harus membuat objek class tersebut. Berikut ini contoh penggunaannya.

```
1 package ddp_09;
2
3 public class Manusia {
4
5     public static void mahasiswa(String nama, int nim) {
6         System.out.println("Nama Mahasiswa "+nama+" dan NIM "+nim);
7     }
8 }
```

Adapun pemanggilan dari metode tersebut pada class utama adalah sebagai berikut.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Manusia.mahasiswa("Siti", 17);
8         Manusia.mahasiswa("Jhon", 20);
9         Manusia.mahasiswa("Jack", 21);
10
11    }
12 }
```

Bagaimana jika keyword static digunakan pada variabel, berikut ini contoh penggunaanya.

```
1 package ddp_09;
2
3 public class Manusia {
4     public static String nama;
5     public static int usia;
6
7     public static void mahasiswa() {
8         System.out.println("Nama Mahasiswa "+nama+" dan usia "+usia);
9     }
10 }
```

Adapun penggunaannya dalam class utama pada java adalah sebagai berikut.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Manusia.nama = "Siti";
8         Manusia.usia = 17;
9
10        Manusia.mahasiswa();
11
12    }
13 }
```

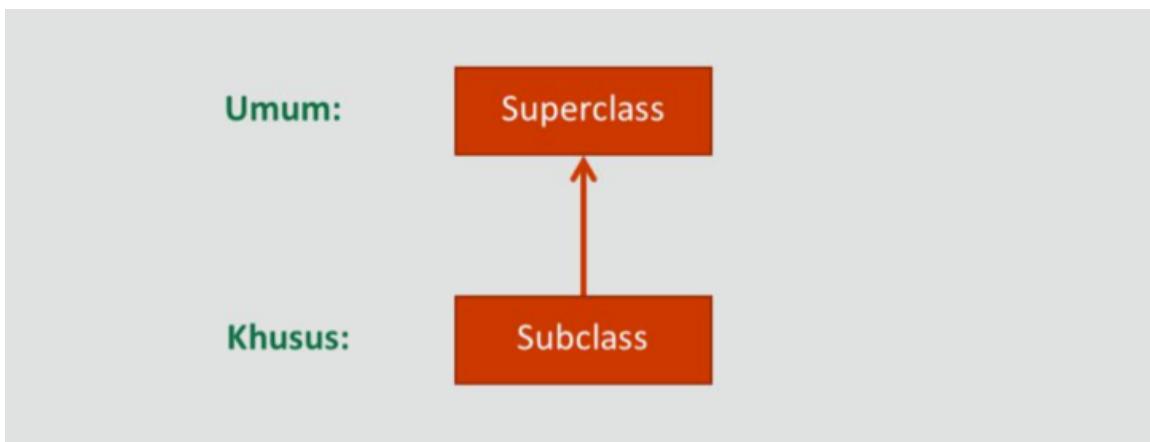
Pada java terdapat nested class, sama halnya dengan pernyataan dan juga perulangan yang mendukung hal ini. Berikut ini contoh penggunaan nested class pada java.

```
1 package ddp_09;
2
3 public class Luar {
4     public String variabel_class_luar; //ubah public ke private
5
6     class Dalam {
7         String variabel_class_dalam;
8
9         void cekClass() {
10             System.out.println(variabel_class_dalam);
11             System.out.println(variabel_class_luar);
12         }
13     }
14 }
```

Adapun cara untuk menggunakan pada class utama adalah sebagai berikut.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Luar l = new Luar();
8         l.variabel_class_luar = "Variabel Class Luar"; // ->harus public
9         Luar.Dalam m = l.new Dalam();
10
11         m.variabel_class_dalam = "Variabel Class Dalam";
12
13         m.cekClass();
14     }
15 }
```

Dalam pemrograman berbasis objek, terdapat konsep yang paling sering digunakan dan konsep ini digunakan untuk memanfaatkan “*code reuse*” untuk menghindari duplikasi kode, konsep ini disebut dengan inheritance (pewarisan). Berikut ini merupakan gambaran dari konsep inheritance.



Superclass	Subclass
Kelas yang lebih umum tempat kelas lain mewarisi metode dan datanya.	Semakin spesifik kelas yang diturunkan atau diwarisi dari kelas lain (superclass).

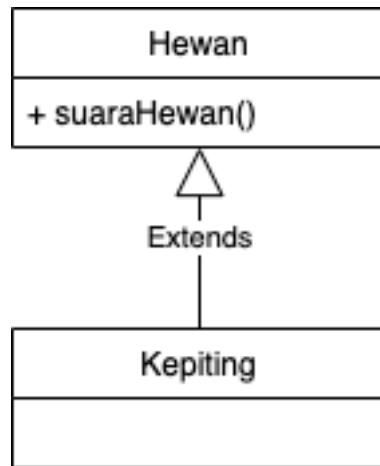
Contoh penerapan dari konsep inheritance seperti ditunjukkan pada gambar berikut.



Gambar di atas menunjukkan bahwa kepiting merupakan subclass sedangkan hewan adalah superclass. Ciri lain untuk menunjukkan suatu inheritance adalah adanya kata kunci extends pada class subclass.

```
public class Rectangle extends Shape
{
    //code
} //end class Rectangle
```

Contoh tersebut menunjukkan bahwa Class Rectangle merupakan subclass dari Class Shape atau turunan dari Class Shape. Adapun contoh implementasi inheritance seperti berikut ini.



#### Class Hewan

```
1 package ddp_09;
2
3 public class Hewan {
4
5     protected void suaraHewan(String suara) {
6         System.out.println("Suaranya: "+suara);
7     }
8
9 }
```

#### Class Kepiting

```
1 package ddp_09;
2
3 public class Kepiting extends Hewan {
4
5 }
```

Setelah membuat super class dan subclass seperti di atas, lakukan pemanggilan metode suaraHewan pada class utama seperti berikut.

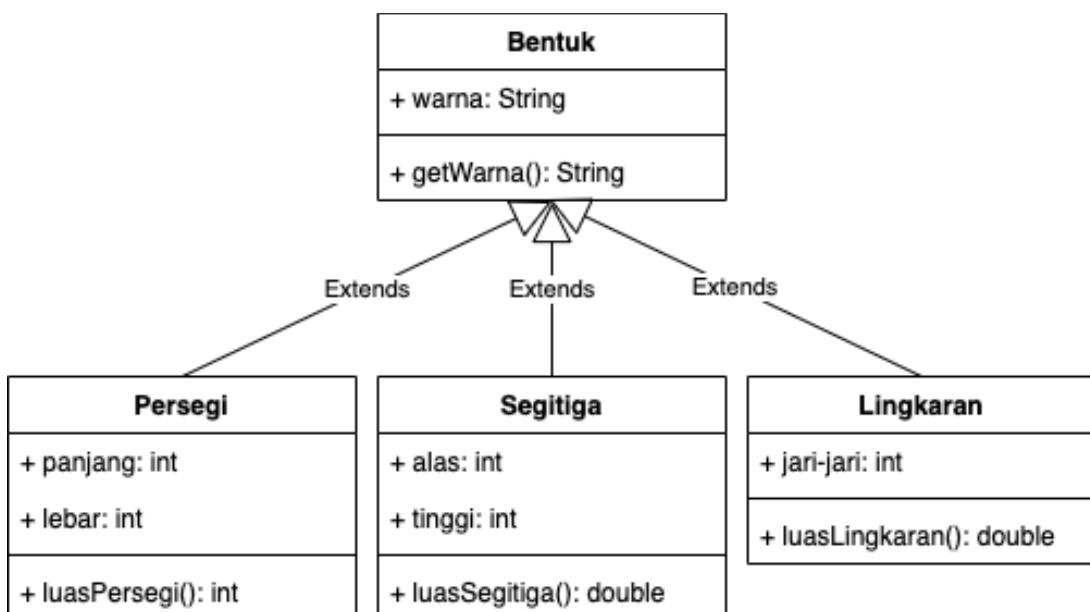
```

1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Hewan h = new Hewan();
8         Kepiting k = new Kepiting();
9
10        h.suaraHewan("tuktukk");
11        k.suaraHewan("tuktuk");
12
13    }
14 }
```

Setelah memahami konsep dari inheritance, terdapat beberapa manfaat dari konsep ini diantaranya adalah:

- a. Memanfaatkan “code reuse”
- b. Tidak perlu menuliskan kode yang sama kembali
- c. Memberi akses ke data yang ada pada super class
- d. Meminimalisir terjadinya “bug”

Konsep dasar yang juga perlu dipahami agar lebih mudah memahami inheritance adalah UML (Unified Modeling Language) dalam bentuk class diagram seperti contoh di bawah ini.



### Class Bentuk

```
1 package ddp_09;
2
3 public class Bentuk {
4
5     String warna;
6
7     Bentuk(String w) {
8         this.warna = w;
9     }
10
11     protected String getWarna() {
12         return this.warna;
13     }
14
15 }
```

### Class Persegi

```
1 package ddp_09;
2
3 public class Persegi extends Bentuk {
4     int panjang;
5     int lebar;
6
7     Persegi(String w, int p, int l) {
8         super(w);
9         this.panjang = p;
10        this.lebar = l;
11    }
12
13    int luasPersegi() {
14        return this.lebar*this.panjang;
15    }
16 }
```

### Class Segitiga

```
1 package ddp_09;
2
3 public class Segitiga extends Bentuk {
4     int alas;
5     int tinggi;
6
7     Segitiga(String w, int a, int t) {
8         super(w);
9         this.alas = a;
10        this.tinggi = t;
11    }
12
13    double luasSegitiga() {
14        return (0.5*this.alas*this.tinggi);
15    }
16
17 }
```

## Class Lingkaran

```
1 package ddp_09;
2
3 public class Lingkaran extends Bentuk {
4     int jari_jari;
5
6     Lingkaran(String w, int jj) {
7         super(w);
8         this.jari_jari = jj;
9     }
10
11     double luasLingkaran() {
12         return (3.14*this.jari_jari*this.jari_jari);
13     }
14
15 }
```

Kemudian lakukan pemanggilan pada class utama untuk melihat hasil dari setiap class turunan dan class induk.

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         Persegi p = new Persegi("Merah", 9, 5);
8         System.out.println(p.getWarna());
9         System.out.println(p.luasPersegi());
10
11         Segitiga s = new Segitiga("Kuning", 2, 3);
12         System.out.println(s.getWarna());
13         System.out.println(s.luasSegitiga());
14
15         Lingkaran l = new Lingkaran("Hijau", 4);
16         System.out.println(l.getWarna());
17         System.out.println(l.luasLingkaran());
18
19         Bentuk b = new Bentuk("Jingga");
20         System.out.println(b.getWarna());
21
22     }
23 }
```

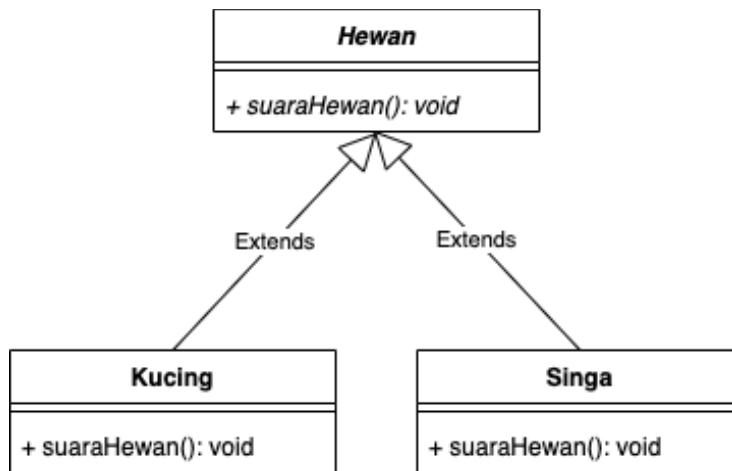
Konsep fundamental lainnya dalam pemrograman berorientasi objek adalah enkapsulasi (encapsulation). Enkapsulasi berarti membungkus data dengan tujuan untuk menghindari data dari kerusakan atau modifikasi dari pengguna lain. Adapun cara untuk membungkus atau menjaga data ini, yaitu dengan menggunakan access modifier seperti

public, private, protected dan default. Lakukan pengujian dengan mengubah metode getWarna pada class induk dengan private dan perhatikan pada class utama, metode getWarna tidak ditemukan. Kemudian pada class induk tambahkan private pada konstruktor, sehingga subclass lainnya tidak dapat menemukan konstruktor pada class induk.

Polimorfisme berarti memiliki banyak bentuk, dalam bahasa pemrograman java, terdapat dua jenis polimorfisme, yaitu statis (overloading) dan dinamis (overriding). Perbedaannya pada metode overloading berarti terdapat beberapa metode dengan nama yang sama dalam satu class seperti berikut ini.

```
1 package ddp_09;
2
3 public class Utama {
4
5     static void kelilingPersegi(int sisi) {
6         System.out.println(sisi*4);
7     }
8
9     static void kelilingPersegi(double sisi) {
10        System.out.println(sisi*4);
11    }
12
13     public static void main(String[] args) {
14
15         Utama.kelilingPersegi(2);
16         Utama.kelilingPersegi(2.0);
17
18     }
19 }
```

Selanjutnya bagaimana dengan polimorfisme dinamis atau overriding, untuk membuat overriding harus menggunakan inheritance dengan memanfaatkan metode abstract. Dalam hal ini subclass menggunakan metode yang ada pada super class seperti berikut ini.



### Class Hewan

```

1 package ddp_09;
2
3 abstract class Hewan {
4
5     protected abstract void suaraHewan();
6
7 }

```

### Class Kucing

```

1 package ddp_09;
2
3 public class Kucing extends Hewan {
4
5     @Override
6     protected void suaraHewan() {
7         System.out.println("Meong...");
8     }
9
10 }

```

### Class Singa

```

1 package ddp_09;
2
3 public class Singa extends Hewan {
4
5     @Override
6     protected void suaraHewan() {
7         System.out.println("Roar...");
8     }
9 }

```

Kata kunci final pada java digunakan untuk mencegah class, properti atau metode ditimpa nilainya. Keyword ini juga masuk dalam modifier pada java yaitu untuk memodifikasi perilaku default dari sebuah class, properti atau metode.

#### Contoh pada Variabel

```
1 package ddp_09;
2
3 public class Utama {
4
5     public static void main(String[] args) {
6
7         final double PI = 3.14;
8 //         PI = 3.11;
9         System.out.println(PI);
10
11     }
12 }
```

#### Contoh pada Metode dan Class

```
1 package ddp_09;
2
3 final class Hewan {
4
5     final static boolean makhlukHidupkah() {
6         return true;
7     }
8 }
9
```

```
1 package ddp_09;
2
3 public class Kucing extends Hewan {
4
5 }
6
```

Class final tidak dapat diturunkan dibuktikan dengan class Hewan yang menggunakan kata kunci final tidak dapat diturunkan pada class Kucing, ini artinya data yang ada pada class Hewan bersifat final atau tidak dapat diubah lagi.

~ SELESAI ~