**MODUL PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK**



# Disusun oleh: Tim Dosen Pemrograman Berorientasi Objek

**PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS TEKNIK UNIVERSITAS PALANGKA**

# RAYA

**2**

# 0

**2**

# 3

TATA TERTIB DAN TATA LAKSANA PRAKTIKUM JURUSAN/PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS TEKNIK UPR

# TATA TERTIB

|  |  |
| --- | --- |
| 1  . | Praktikan **WAJIB** mengikuti semua modul praktikum. |
| 2  . | Praktikan hanya boleh tidak mengikuti praktikum **1 (satu) kali DENGAN ATAU TANPA surat izin** dari jumlah pertemuan praktikum. |
| 3  . | Praktikan yang berhalangan mengikuti praktikum, diwajibkan melaporkan ke asisten praktikum untuk menentukan jadwal praktikum sebagai pengganti jadwal yang berhalangan. |
| 4  . | Praktikan yang lebih dari 1 (satu) kali tidak mengikuti praktikum, tidak diperbolehkan untuk mengikuti praktikum untuk modul-modul praktikum selanjutnya dan **NILAI AKHIR PRAKTIKUM** adalah **NOL**. |
| 5  . | Praktikan wajib hadir 15 menit sebelum praktikum dimulai. |
| 6  . | Tidak diperbolehkan saling bekerjasama. |
| 7  . | Dilarang menggunakan kaos oblong dan sandal jepit selama praktikum. Bagi yang melanggar poin ini, tidak diperbolehkan mengikuti praktikum. |

**TATA LAKSANA**

|  |  |
| --- | --- |
| 1. | Sebelum praktikum dimulai, setiap praktikan wajib mengumpulkan **LAPORAN RENCANA PRAKTIKUM (Tulis Tangan)** dan **LAPORAN HASIL**  **PRAKTIKUM** modul sebelumnya. |
| 2. | Jika praktikan tidak melaksanakan tata laksana poin 1, maka  **TIDAK DIPERBOLEHKAN MENGIKUTI PRAKTIKUM**. |
| 3. | Setiap modul praktikum, akan dilakukan ***Pre-test***. |
| 4. | Pada saat praktikum hanya membawa alat tulis, modul dan Laporan. HP, Laptop & USB disimpan dan tas letakan didepan. |
| 5. | Pada saat praktikum tidak diizinkan mengakses Internet. |
| 6. | Format laporan meliputi:  **Laporan Rencana Praktikum** :  Format A4  Tujuan Praktikum dari modul yang akan dilaksanakan Hal-hal yang akan dilakukan selama praktikum.  **Laporan Hasil Praktikum** :  Halaman Depan  Bab I. Tujuan dan Landasan Teori Bab II. Pembahasan |

Bab III. Kesimpulan Daftar Pustaka

Lampiran (disertai Laporan Rencana Praktikum)

7. Format Penulisan Laporan Hasil Praktikum Spasi : 1,5

Font : Times New Roman Font Size : 12

Margins : Top 3 cm, Left 4 cm, Right 3 cm, Bottom 4 cm Kertas : A4

|  |  |  |
| --- | --- | --- |
| 8. | Penilaian Laporan Hasil Praktikum : |  |
|  | Bab I. Tujuan dan LandasanTeori | Nilai 10 |
|  | Bab II. Pembahasan | Nilai 60 |
|  | Bab III. Kesimpulan | Nilai 20 |
|  | Daftar Pustaka Lampiran | Nilai 5 Nilai 5 |
|  | **Total** | **100** |

1. Praktikan yang mengabaikan format penulisan poin 6, nilai akan dikurangi 5 untuk setiap kesalahan.
2. PenilaianAkhirPraktikum :

*Pre-Test* : 10%

Praktikum : 50% Laporan Praktikum : 15% Responsi : 25%

**Total**

**100%**

1. PenilaianAkhir Mata Kuliah: Kuliah, meliputi:

Tugas : 20%

UTS : 30%

Praktikum : 50%

# Total : 50%

UAS : 50%

# Total : 100%

**MODUL I**

# PENGENALAN OOP MENGGUNAKAN ALICE 3

* 1. **Tujuan**
     1. Mahasiswa mampu membuat Objek dan mengimplementasikan method/behavior tiap objek.
     2. Mahasiswa mampu memahami konsep OOP melalui pemrograman IDE Alice.

# Landasan Teori

Alice adalah bahasa pemrograman pendidikan berbasis objek dengan lingkungan pengembangan terintegrasi (IDE). Alice menggunakan lingkungan drag and drop untuk membuat animasi komputer menggunakan model 3D

# Initial Scene Components

Initial Scene adalah titik awal animasi anda, yang mana memiliki 3 komponen :

* + - 1. Background template, seperti langit, tanah, cahaya dan lain-lain.
      2. Non-moving object, objek yang menggambarkan suasana/adegan.
      3. Moving objects, objek yang bergerak dan mampu memberikan aksi.

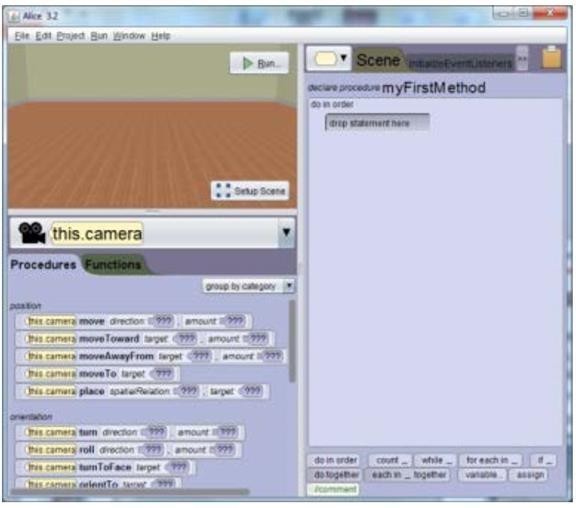


Gambar 1.1 Komponen Pada Scene.

# Navigasi Antar Editor

Alice menyediakan dua editor ruang kerja yang berbeda. Kedua editor tersebut adalah :

1. Code Editor, adalah tempat Anda menambahkan instruksi pemrograman untuk memprogram animasi Anda.



Gambar 1.2 Jendela Code Editor

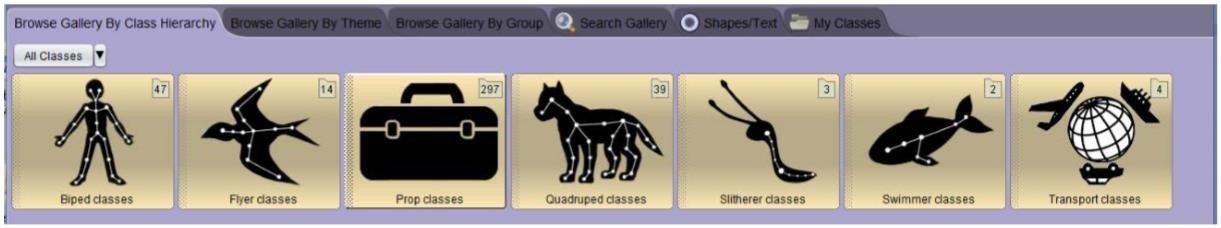
1. Scene Editor, pada Scene Editor anda dapat:
   * memilih objek dari galeri untuk ditambahkan ke adegan.
   * Posisikan objek dalam adegan menggunakan palet Handles.
   * Edit properti objek menggunakan panel Properties.
   * Jalankan animasi setelah pernyataan pemrograman ditambahkan ke editor Kode.



Gambar 1.2 Jendela Scene Editor

# Memilih Kelas

Tab Class Hierarchy mengelompokkan objek menurut jenis mobilitas (biped, flyer, dll.). Kelas berisi instruksi yang menentukan penampilan dan pergerakan suatu objek. Semua objek di dalam kelas memiliki properti yang sama. Kelas tersebut memberikan instruksi kepada Alice 3 untuk membuat dan menampilkan objek ketika ditambahkan ke adegan Anda.



Gambar 1.3 Kelas yang tersedia di Alice 3

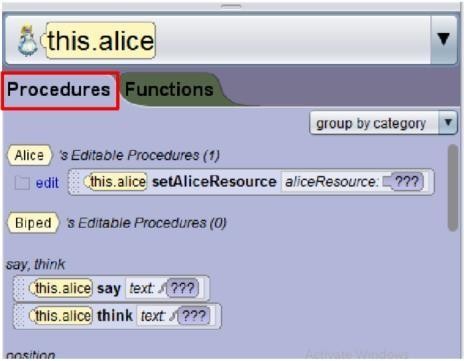
# Menambahkan Objek

Di scene editor, menambahkan objek (instance of class) ke adegan bisa dilakukan dengan dua cara:

* + - 1. Drag and Drop, Seret objek dari galeri ke adegan dengan mouse Anda.
      2. Klik objek, dan selesaikan kotak dialog, kemudian biarkan Alice 3 menambahkan objek tersebut ke tengah layar.

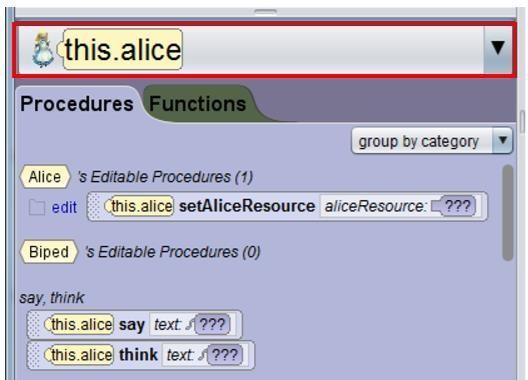
# Method panel

Tab Prosedur, yang terletak di dalam Panel Metode di code editor, menampilkan metode yang ditentukan sebelumnya untuk contoh objek yang dipilih, serta metode yang ditentukan untuk kelas objek.



Gambar 1.4 Panel Method

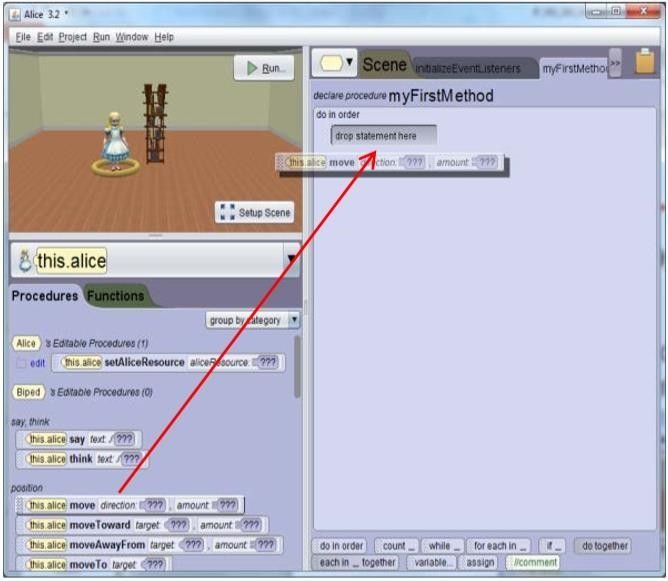
Menu Instance di Panel Metode, Menu instance ditampilkan di atas tab Prosedur. Drop down Segitiga penunjuk ke bawah di sisi kanan menu menunjukkan bahwa menu turun saat dipilih.



Gambar 1.5 Menu Instance di Panel Metode

# Membuat Instruksi Pemrograman

Dari Panel Metode, klik dan seret instruksi pemrograman yang diinginkan ke myFirstMethodtab editor Kode.



Gambar 1.6 Memilih Intruksi Program

Setelah Anda menyeret instruksi pemrograman ke myFirstMethodtab, gunakan menu berjenjang untuk memilih nilai untuk setiap argumen yang digunakan dalam metode.



Gambar 1.7 Menentukan Value untuk Argumen Method

# Jenis Argumen Prosedur

Beberapa jenis/tipe argumen :

* Direction
* Amount
* Duration
* Text

Alice 3 mengenali berapa banyak argumen yang dibutuhkan untuk setiap instruksi pemrograman. Ini memberi Anda jumlah yang benar dari menu berjenjang untuk menentukan nilai untuk masing-masing argumen tersebut.

# Praktikum

1. Buat proyek baru menggunakan template gurun. Tambahkan objek singa betina dan tiga objek batu besar pada scene.



Gambar 1.8 Scene Praktikum 1

* 1. Gunakan fungsi untuk memprogram singa betina agar bergerak tepat di samping setiap objek batu besar; bukan menembus/melewatinya.
  2. b. Simpan proyek.

1. Buat proyek baru menggunakan template berumput. Tambahkan tiga pohon dan satu burung ke tempat kejadian. Posisikan burung di salah satu pohon. Pastikan jarak pohon setidaknya 10 meter satu sama lain.



Gambar 1.9 Scene Praktikum 2

* 1. Gunakan fungsi untuk memprogram burung agar terbang dari satu pohon ke pohon berikutnya, mendarat di pohon tetapi tidak melewatinya.

*Petunjuk*: Nyatakan prosedur untuk gerakan terbang lebih dulu. Kemudian, buatlah burung mengepakkan sayapnya sekaligus terbang jarak ke pohon.

* 1. Tetapkan burung sebagai setVehicle untuk kamera, sehingga kamera mengikuti burung saat terbang di sekitar pemandangan.
  2. Simpan proyek.

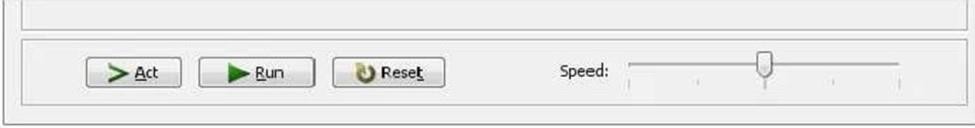
# MODUL II PENERAPAN OOP PADA GREENFOOT

* 1. **Tujuan**
     1. Mahasiswa mampu memahami konsep OOP melalui pemrograman IDE Greenfoot.
     2. Mahasiswa mampu meng-implementasikan OOP pada scenario/adegan yang dibuat.

# Landasan Teori

* + 1. **Scenario**

Scenario adalah sebuah permainan atau simulasi yang diimplementasikan di Greenfoot. Untuk mengeksekusi suatu scenario, dapat menggunakan control eksekusi yang terdiri dari : a. Act, menjalankan semua aksi di skenario satu kali

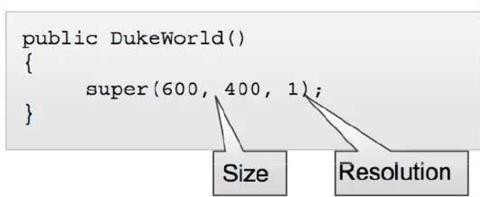
* + 1. Run/pause, menjalankan semua aksi di skenario berulang-ulang sampai tombol Pause di-klik
    2. Reset, memberhentikan sebentar skenario atau mengulang skenario kembali ke posisi awal
    3. Speed, menjalankan aksi lebih cepat atau lambat

Gambar 2.1 Kontrol Eksekusi

# Superclass

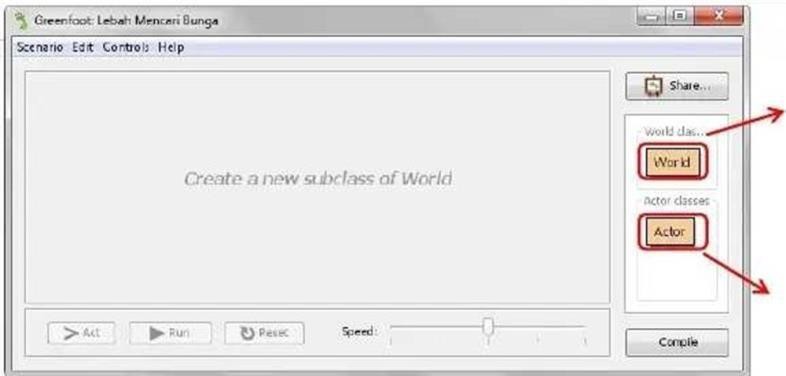
Greenfoot memilki beberapa tipe superclass, yaitu :

* + - 1. World, memegang subclass yang menyediakan background gambar untuk skenario world dan mendefinisikan ukuran dan resolusi world



Gambar 2.2 Merubah Ukuran World

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

* + - 1. Actor, memegang subclass-subclass yang menghasilkan instances yang beraksi di skenario.

Superclass World

Gambar 2.3 Superclass World dan Actor

Superclass Actor

# Subclass

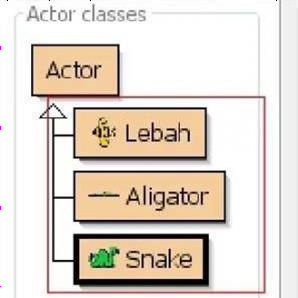
Subclass adalah kelas tururan dari superclass

* + - 1. Menurunkan semua properti superclass Actor, contohnya mendefiniskan lebih awal aksi-aksi yang dapat dilakukan oleh Aktor subclass
      2. Dapat memiliki properti baru yang dibuat programmer khusus untuk subclass seperti gambar dan aksi
      3. Properti subclass

Subclass memiliki hubungan “is-a” terhadap Superclass contoh : Lebah is a subclass of the Actor superclass

Properti dapat dimodifikasi (seperti nama Class, gambar yang ditampilkan, atau aksi-aksi yang dilakukan).

Panah pada hirarki Class menunjukkan hubungan antara subclass dengan superclass



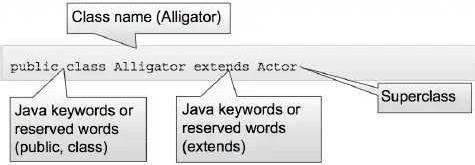
Gambar 2.4 Subclass Actor

# Class Definition

Class Definition mencakup:

Kata kunci “class” untuk mendefinisikan class

Kata kunci “extends” untuk mendefinisikan pewarisan Nama class (subclass)

Nama superclass yang diwarisi subclass

Gambar 2.5 Class Definition Isi

dari sebuah class:

Variabel (atau field) : menyimpan data dalam object/instance

Sebuah variabel memiliki syntax penulisan sebagai berikut: #Access modifier*: public/protected/default/private*

# Tipe data yang disimpan # Nama variabel Contoh

:



Access modifier Tipe data Nama variabel

# Object / Instances

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

# Method

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

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

Return type : Menspesifikasikan data yang dikembalikan oleh method

Method name : Mendeskripsikan pekerjaannya si method Parameter List : Informasi yang masuk kedalam panggilan method

Gambar 2.7 Contoh Method

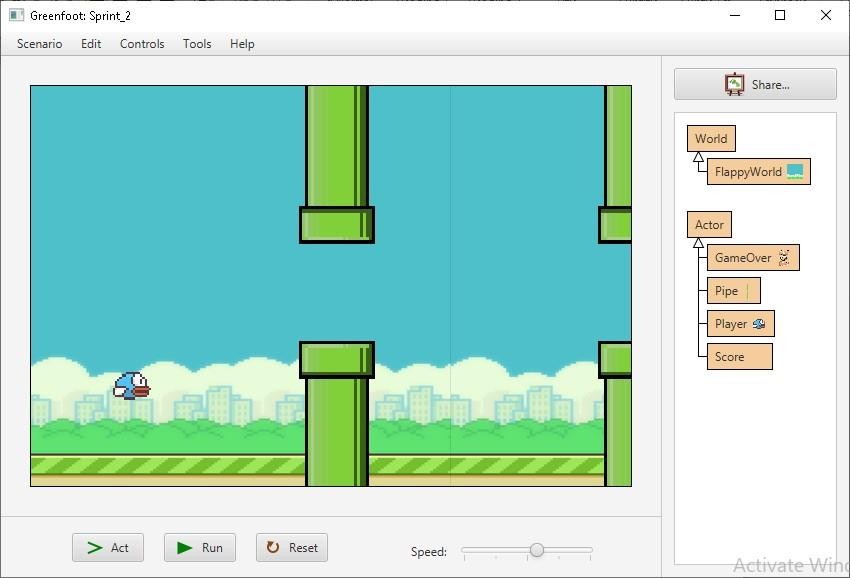
# Praktikum

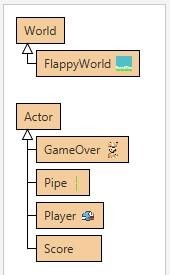
* 1. **Tugas**

Buatalah sebuah permainan yang menyerupai game legendaris “Flappy Bird”. Logika Permainan :

1. Jika Flappy bird menabrak pipa, GameOver
2. Jika Flappy bird menyentuh edge(sudut bawah layar), GameOver
3. Jika Flappy bird melewati pipa dan dead!=TRUE, SCORE +1
4. Jika Space button pressed, Flappy bird terbang(flap)

Untuk keperluan resource seperti img silahkan didownload pada link berikut : [www.mrferrante.com/flappybird/flappy\_bird\_images.zip](http://www.mrferrante.com/flappybird/flappy_bird_images.zip)



Gambar tampilan scenario game flappy bird

Gambar Class dan Object yang dibuat

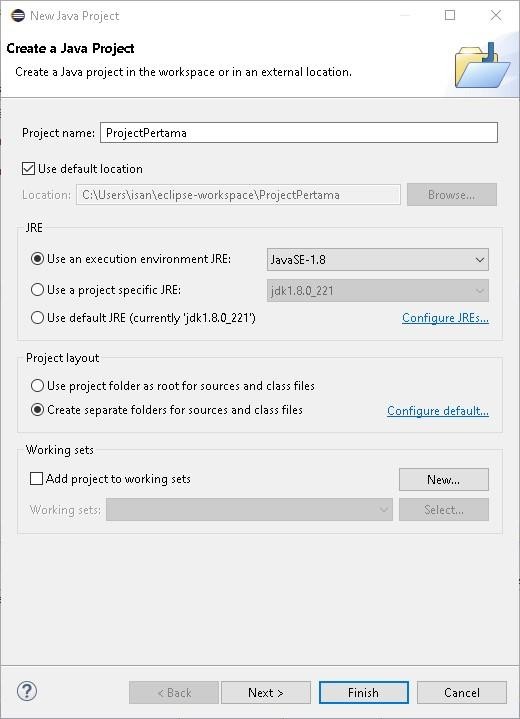
# MODUL III

**MEMULAI JAVA OOP DENGAN ECLIPSE**

# TUJUAN

* + 1. Mahasiswa mampu membuat Program Java serta menerapkan konsep OOP menggunakan aplikasi Eclipse.
    2. Mahasiswa mampu membedakan *Class Driver* dan *Class Object.*

# LANDASAN TEORI

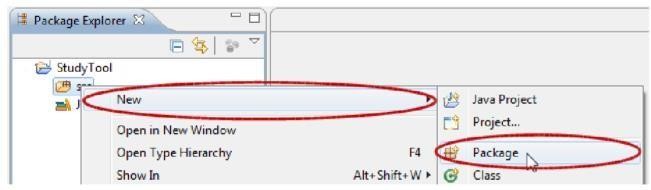
* + 1. **Membuat Project pada Eclipse**
       1. Pilih File → New → Java Project\*.
       2. Masukan nama project (CamelCase) dan klik finish.

# Gambar 3.1 Membuat Java Project

* + - 1. Dengan klik finish value project ter-set default.

# Membuat Package

* + - 1. Select src folder pada package view.
      2. Klik kanan src folder dan pilih New → Package.



Gambar 3.2 Membuat Package.

* + - 1. Berikan nama package sama seperti nama project namun menggunakan gaya penulisan lowerCamelCase.
      2. Klik finish.

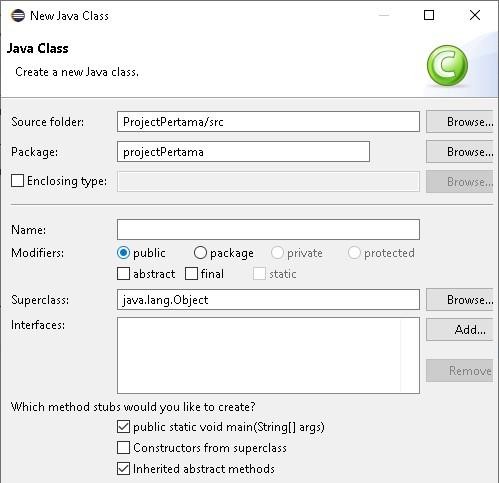
Terdapat dua cara dalam menggunakan package, yaitu:

* + - * 1. Apabila class yang menggunakan berada dalam package yang sama dengan class yang digunakan, maka tidak perlu mengimpor package.
        2. Apabila class yang menggunakan berada dalam package yang berbeda dengan class yang digunakan, maka pada awal source code harus dicantumkan perintah :

import namaPackage.namaClass; //atau import namaPackage.\*;

# Membuat Class

* + - 1. Buat class pada folder src dengan cara klik kanan kemudian pilih New → Class.
      2. Masukan nama class kemudian pilih untuk membuat main method didalam class.



Gambar 3.3 Membuat Class

# Acces Modifiers

Ada 4 hak akses yang tersedia, yaitu default, public, protected, private.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aksesabilitas | Public | Protected | Private | Default |
| Class Package Subclass Global | Ya | Ya | Ya | Ya |
| Ya | Ya | Tidak | Ya |
| Ya | Ya | Tidak | Tidak |
| Ya | Tidak | Tidak | Tidak |

# Konstruktor

Konstruktor adalah method unik di java, karena metode ini membuat instance kelas(objek). Konstruktor selalu memiliki nama yang sama dengan kelas dan tidak mendeklarasikan tipe kembalian.Konstruktor dipanggil menggunakan kata kunci baru.

* + - 1. Konstruktor tanpa parameter

Jika Anda membuat konstruktor tanpa parameter (tanda kurung kosong), Anda dapat mengosongkan konten konstruktor (antara

{dan}). Ini disebut konstruktor default, dan sama dengan konstruktor yang disediakan Java jika Anda tidak mendeklarasikannya.

* + - 1. Konstruktor dengan parameter

Jika Anda membuat konstruktor dengan parameter (tanda kurung TIDAK kosong), Anda juga akan menginisialisasi variabel antara

{dan}. Konstruktor ini akan menginisialisasi variabel kelas dengan nilai yang dikirim dari kelas driver utama.

# Main Method

Untuk menjalankan program Java, harus menentukan metode utama dalam Driver Class. Metode utama secara otomatis dipanggil saat kelas dipanggil. Ingatlah untuk memberi nama file tersebut sama dengan kelasnya.

# Accessor and Mutator Methods

Accessor and Mutator Methods merupakan hal yang umum untuk membuat sekumpulan metode yang memanipulasi dan mengambil nilai data kelas.

* + - 1. Accessors (getter): Metode yang mengembalikan (mendapatkan) nilai dari setiap variabel kelas.
      2. Mutators (setter): Metode yang mengubah (set) nilai setiap variabel kelas.

public int getStudentId() { return studentId;

}

public void setStudentId(int studentId)

{ this.studentId = studentId; }

# PRAKTIKUM

* + 1. **Students.java**

**package** com.example.domain;

**public class** Students {

**private int** studentId; **private** String name; **private** String ssn; **private double** gpa;

**public final int** SCHCODE = 34958;

**public** Students() {

}

**public** Students(**int** studentId, String name, String ssn, **double**

gpa){

**this**.studentId=studentId; **this**.name=name;

**this**.ssn=ssn;

**this**.gpa=gpa;

}

**public int** getStudentId() {

**return** studentId;

}

**public** String getName() {

**return** name;

}

**public** String getSsn() {

**return** ssn;

}

**public double** getGpa() {

**return** gpa;

}

**public void** setStudentId(**int** studentId) {

|  |  |
| --- | --- |
| } | **this**.studentId = studentId;  }  **public void** setStudentName(String name) {  **this**.name=name;  }  **public void** setSSN(String ssn) {  **this**.ssn=ssn;  }  **public void** setGPA(**double** gpa) {  **this**.gpa=gpa;  } **public** String toString()  {  String s1 = "";  s1 = "Student Id: " + getStudentId() +  "Student Name: " + getName() + "Student SSN: " + getSsn() +  "Student GPA: " + getGpa();  **return** s1;  } |

# StudentTester.java

**package** com.example.domain;

**public class** StudentTester {

**public static void** main(String[] args) {

Students s1 = **new** Students(123, "Mary Smith", "999- 99-9999", 3.4);

System.***out***.println(s1); Students s2 = **new** Students(); s2.setStudentId(124); s2.setStudentName("John Jacoby");

s2.setSSN("123-45-6789"); s2.setGPA(4.0);

System.***out***.println(s2);

}

}

* 1. **TUGAS**

Membuat proyek inventori:

* + 1. Untuk bagian pertama proyek, Anda diminta untuk memikirkan tentang apa yang akan disimpan oleh sistem inventaris Anda.
       1. Pikirkan produk tertentu yang dapat disimpan dalam inventaris (misalnya, produk di rumah, sekolah, atau tempat kerja Anda. Tulis

daftar setidaknya 6 produk yang ingin Anda simpan di sistem Anda, proyek ini dapat digunakan untuk menyimpan berbagai macam produk.

* + - 1. Untuk setiap produk yang Anda identifikasi, buat dalam tabel seperti berikut ini:

|  |  |
| --- | --- |
| Atribut | Sample Data |
| Nama produk (nilai yang akan mengidentifikasi produk di  sistem Anda). | Rayzen 5 5600X |
| Harga (nilai ini memegang harga setiap barang yang akan dijual). | 5000000 |
| Jumlah unit yang tersedia (nilai ini akan menyimpan berapa banyak dari setiap item produk yang saat ini tersedia). | 5 |
| Nomor item (digunakan untuk mengidentifikasi produk secara  unik di sistem Anda). | 1 |

Tabel ini memberi Anda pemahaman tentang jenis data yang ingin Anda simpan untuk atribut setiap produk. Hal ini berguna sehingga Anda memiliki pemahaman yang jelas tentang data yang akan Anda tangani!

* + 1. Langkah selanjutnya adalah memikirkan tipe data yang akan Anda gunakan untuk menyimpan nilai di sistem Anda.

|  |  |  |
| --- | --- | --- |
| Atribut | Sample Data | Tipe Data |
| Nama produk | Rayzen 5 5600X | String |
| Harga | 5000000 | double |
| Jumlah | 5 | int |
| Nomor Item | 1 | int |

* + 1. Buka Eclipse dan buat proyek bernama inventori
    2. Buat kelas objek bernama Product
    3. Tambahkan variabel dengan menggunakan tipe data yang sudah Anda identifikasi di tugas 2: a. nomor barang

1. nama produk
2. jumlah unit yang tersedia
3. harga setiap unit
   * 1. Buat 2 Konstruktor
        1. Sebuah konstruktor default tanpa parameter yang memungkinkan compiler untuk menginisialisasi field ke nilai defaultnya. Tambahkan komentar di atas konstruktor Anda yang menjelaskan tujuan kode.
        2. Konstruktor dengan parameter untuk keempat variabel kelas sehingga dapat diinisialisasi dengan nilai dari kelas driver.
     2. Buat metode getter / accessor dan setter / mutator untuk masing-masing dari empat variabel instance. Tambahkan komentar di atas mereka untuk menjelaskan tujuan mereka.
     3. Buat sebuah metode untuk menampilkan nilai tiap variabel. Format :

Nomor : 1

Nama : Ryzen 5 5600x

Stok : 5

Harga : 5000000

* + 1. Buat kelas utama Java yang disebut ProductTester.
    2. Membuat dan menginisialisasi enam objek Produk berdasarkan daftar yang Anda buat di tugas 1.
       1. Dua dari Produk harus dibuat menggunakan konstruktor default.
       2. Empat lainnya harus dibuat dengan memberikan nilai untuk argumen yang cocok dengan parameter konstruktor.
    3. Menggunakan kelas ProductTester, tampilkan detail setiap produk ke konsol.
    4. Simpan proyek Anda.

# MODUL IV

**Scanner, Kondisional Statement dan Arrays**

# TUJUAN

* + 1. Mahasiswa mampu menerapkan scanner kedalam program yang dibuat.
    2. Mahasiswa mampu memahami dan menerapkan statement kondisi dan kontrol.
    3. Mahasiswa mampu menerapkan array kedalam program yang dibuat.

# LANDASAN TEORI

* + 1. **Scanner**

Inputasi melalui keyboard menggunakan Scanner membutuhkan pernyataan import berikut ***import java.util.Scanner***. Untuk membaca masukan yang telah dimasukkan pengguna, gunakan objek Java scanner. Cara untuk menginisialisasi Scanner, tulis :

Scanner in = new Scanner(System.in);

Berikut ini adalah method-method dari class Scanner yang sering digunakan:

|  |  |
| --- | --- |
| **Method** | **Penjelasan** |
| next() | Mengembalikan token berikutnya dari scanner |
| nextLine() | Memindahkan scanner ke baris berikutnya dan mengembalikan nilai dalam string |
| nextByte() | Mengscan token berikutnya sebagai suatu byte |
| nextShort() | Mengscan token berikutnya sebagai suatu nilai short |
| nextInt() | Mengscan token berikutnya sebagai suatu nilai int |
| nextLong() | Mengscan token berikutnya sebagai suatu nilai long |
| nextFloat() | Mengscan token berikutnya sebagai suatu nilai float |
| nextDouble() | Mengscan token berikutnya sebagai suatu nilai double |
| hasNext() | Mengembalikan nilai true jika scanner memiliki token lainnya dalam input |
| Close() | Untuk menutup scanner |

|  |  |
| --- | --- |
| Scanner useDelimiter (String pattern) | Method ini menetapkan pola pemisah scanner ke pola yang dibangun dari String yang ditentukan. |

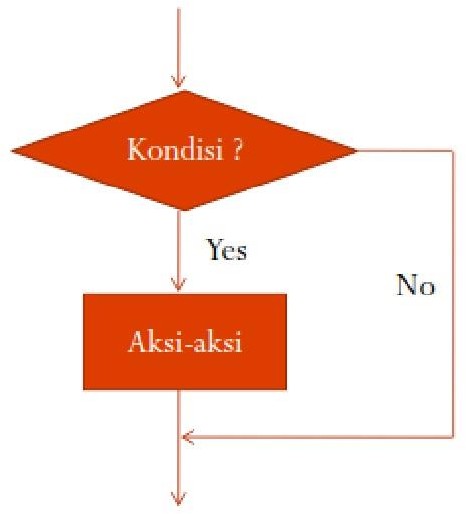
# Conditional dan Control Statement

Statement kontrol dan kondisional digunakan untuk mengatur jalannya alur program sesuai degan yang kita inginkan. Statemenstatement ini dikategorikan kedalam tiga jenis yaitu: pemilihan, pengulangan, dan statement peloncatan.

* + - 1. Pemilihan

Pemilihan adalah suatu keadaan dimana pernyataan dapat dieksekusi apabila suatu kondisi memenuhi syarat untuk mengerjakan pernyataan tersebut. a. Statement if

Untuk membandingkan kondisi pertama, yang berisi sebuah kondisi boolean, jadi bernilai true atau false. Statement if akan dijalankan jika kondisi bernilai true.



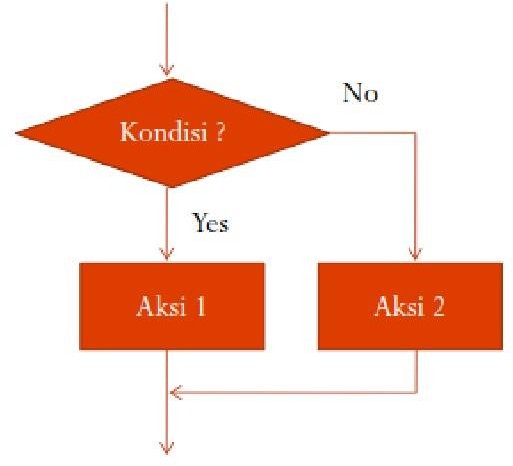
Gambar 2.1 flow chart statement if

If (kondisi) {

//statement yang akan dieksekusi }

b. Statement if ... else

Jenis pemilihan ini relatif lebih kompleks apabila dibandingan dengan jenis pertama diatas. Pada bentuk ini, program menyediakan kondisi tambahan untuk menangani kejadian yang kondisi pertamanya tidak terpenuhi.



Gambar 2.2 flow chart statement if...else

If(kondisi){

//statement akan dieksekusi

}

else if(kondisi2){

//statement dieksekusi jika kondisi2 bernilai True.

}

c. Switch

Statement Switch adalah cara lain untuk mengubah aliran program tergantung pada nilai input. Pernyataan switch menggunakan 3 kata kunci: switch, case, dan default.

Switch: menentukan variabel mana yang akan diuji nilainya. Case: membandingkan nilai variabel sakelar.

Default: ketika input tidak cocok dengan kasus manapun,

kompilator memilih aksi default.

Switch(var){ case : //perintah default :

//perintah

}

* + - 1. Perulangan

Perulangan digunakan untuk melakukan eksekusi terhadap statemen secara berulang sesuai dengan kondisi yang ditentukan.

* + - * 1. For

Perintah akan dijalankan secara terus-menerus selama kondisi dan inisialisasi bernilai true.

For(inisialisasi;kondisi;increment){

//statement yang akan diulang

}

Inisialisasi adalah tipe data dan nilainya. Kondisi adalah kondisi yang akan dibandingkan dengan inisialisasi Increments adalah kondisi yang dijalankan (bisa

increment atau descrement).

* + - * 1. While

Perulangan yang mendifinisikan kondisi diawal blok. Apabila kondisi tidak terpenuhi(bernilai false) maka perulangan pun tidak akan pernah dilakukan.

While(kondisi){

//statemen yang diulang ... Increment/perintah tertentu

}

* + - * 1. Do ... While

Pada struktur while, kondisi ditempatka diawal blok perulangan sedangkan pada struktur do...while kondisi berada diakhir blok.

Do{

//statemen yang diulang...Increment/perintah tertentu

}while(kondisi);

* + - 1. Array

Array adalah sebuah variabel yang bisa menyimpan banyak data dalam satu variabel. Array menggunakan indeks untuk memudahkan akses terhadap data yang disimpannya. Untuk mendeklarasikan sebuah array dapat dituliskan seperti berikut ini :

//cara pertama String[] nama;

// cara kedua String nama[];

//cara ketiga dengan kata kunci new String[] nama = new String[5];

Kurung siku [] digunakan untuk membuat array, kurung siku bisa

diletakkan setelah tipe data atau nama array dan angka 5 dalam kurung artinya batas atau ukuran array-nya.

* + - * 1. Multidimensi Array

Array multidimensi adalah array yang berisi satu atau lebih array. Untuk membuat array dua dimensi, tambahkan setiap array dalam set kurungnya sendiri:

Tipedata [][] namaarray = new tipedata [n][m];

Dimana n adalah jumlah elemen untuk baris dan m adalah

jumlah elemen kolom.

* + - * 1. Array List

Array list merupakan sebuah class yang memungkinkan kita membuat sebuah objek untuk menampung apapun. Contoh Program dengan Array List

import java.util.ArrayList; public class Doraemon { public static void main(String[] args) {

// membuat objek array list

ArrayList kantongAjaib = new ArrayList();

// Mengisi kantong ajaib dengan 5 benda kantongAjaib.add("Senter Pembesar"); kantongAjaib.add(532); kantongAjaib.add("tikus"); kantongAjaib.add(1231234.132); kantongAjaib.add(true);

// menghapus tikus dari kantong ajaib kantongAjaib.remove("tikus"); //

Menampilkan isi kantong ajaib System.out.println(kantongAjaib);

// menampilkan banyak isi kantong ajaib

System.out.println("Kantong ajaib berisi "+ kantongAjaib.size() +" item"); }

}

# Try/Catch Block

Untuk menangani error di Java, digunakan sebuah statement yang bernama try..catch. Statement tersebut digunakan untuk mengurung eksekusi yang menampilkan error dan dapat membuat program tetap berjalan tanpa dihentikan secara langsung. Error yang ditangani oleh try..catch biasa disebut dengan exception.

try{

...code that might cause an exception

}

catch (exception e){

...code to handle the exception }

# PRAKTIKUM

* + 1. **Program Kalkulator Sederhana**

}

}

import java.util.Scanner; public class Section5Example{ public static void main(String[] args){ boolean quit = false;

int num1 = 10, num2 = 6, answer = 0; char operand = '';

Scanner in = new Scanner(System.in); do

{

System.out.println("Please enter a mathematical operand");

String input = in.next(); char operand = input.charAt(0); switch(operand) { case '\*': answer

= num1 \* num2; break;

case '+': answer = num1 + num2; break;

case '-': answer = num1 – num2; break;

case '/': answer = num1 / num2;

default: System.out.println("Invalid input.");

}

System.out.println("Quit? Y/N"); if(in.next().equalsIgnorecase("Y")) quit=true;

}

while(!quit);

# Program Data Karyawan

**Class myInput.java**

package arrayKaryawan; import java.io.\*; public class myInput { public String bacaString(){

BufferedReader bfr=new BufferedReader(new InputStreamReader(System.in),1);

String string =" "; try { string=bfr.readLine();

} catch (IOException ex) { System.out.println(ex);

}

return string;

}

public int bacaInt(){ return Integer.parseInt(bacaString()); } }

# Class dataKaryawan.java

package arrayKaryawan; public class dataKaryawan { private int[] nik=new int[100]; private String[] nama=new String[100]; private int[] gapok=new int[100]; public int n;

public void setNIK(int i,int x){ this.nik[i]= x;

}

public void setNama(int i,String x){ this.nama[i]=x;

}

public void setGapok(int i,int x){ this.gapok[i]=x;

}

public float rerataGapok(int i,int x){ int sum=0; for(i=0;i<this.n;i++){ sum +=this.gapok[i];

}

return sum/this.n;

}

public void setJmlKaryawan(int x){ this.n=x;

}

public void tampilData(){ int i;

System.out.println("================================

=======================");

System.out.println("NIK NAMA GAJI"); System.out.println("================================

=======================");

for(i=0;i<this.n;i++){

System.out.println(String.format("%-5s %-35s Rp %10d", nik[i],nama[i],gapok[i]));

} System.out.println("================================

=======================");

System.out.println("rata=rata gaji pokok dari "+this.n+"karyawan adalah Rp"+this.rerataGapok(i, i)); }

}

**Class mainKaryawan.java**

package arrayKaryawan; public class mainKaryawan { public static void main(String[] args) { int i;

dataKaryawan datakar=new dataKaryawan(); myInput in=new myInput(); System.out.println("berapa jumlah karyawan : "); datakar.setJmlKaryawan(in.bacaInt()); for (i=0;i<datakar.n;i++){

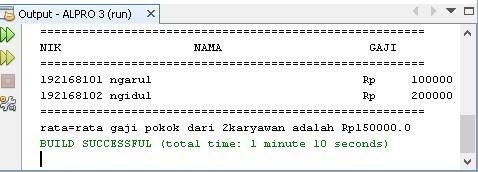
System.out.println("karyawan ke- "+(i+1)); System.out.println("masukan Nama : "); datakar.setNama(i, in.bacaString()); System.out.println("masukan NIK : "); datakar.setNIK(i,in.bacaInt()); System.out.println("masukan gaji pokok : "); datakar.setGapok(i, in.bacaInt());

}

datakar.tampilData();

}

}



Tampilan Output Program Datakaryawan

# TUGAS

1. Sebuah rumah sakit sedang mencari cara untuk mencatat daftar pasien menggunakan java, anda telah dipekerjakan untuk membuat program tersebut menggunakan Bahasa java dan struktur OOP.
2. Tentukan data pasien yang perlu di isikan (minimal 3)
3. Pastikan program dapat terus menginputkan data sebanyak mungkin
4. Rumah sakit ingin fitur pencarian untuk nama pasien
5. Data yang di inputkan di program dapat ditampilkan kembali
6. Program hanya dapat berakhir jika memilih keluar
7. Berikut adalah struktur program Terdapat 3 public class
   * Pasien
   * Rumah Sakit
   * Main
8. Berikut adalah library yang akan digunakan

import java.util.Scanner; import java.util.ArrayList; import java.util.List;

1. Data yang diimputkan berjenis list dengan diisikan elemen sebuah class

private List<nama Class> nama List;

1. Jangan lupa gunakan Konstruktor, Setter, dan Getter
2. Simpan proyek Anda.

# MODUL V

**Inheritance dan Polymorphism**

# TUJUAN

* + 1. Mahasiswa dapat menggunakan teknik pewarisan dan polymorfisme
    2. Mahasiswa mengetahui perbedaan overriding dan overloading.

# LANDASAN TEORI

* + 1. **Pewarisan (Inheritance)**

Inheritance merupakan proses pewarisan data dan method dari suatu class yang telah ada kepada suatu class baru. Class yang mewariskan disebut dengan **superclass / parent class / baseclass**, sedangkan class yang mewarisi (class yang baru) disebut dengan **subclass / child class**

# /derived class.

Subclass tidak dapat mewarisi anggota private dari superclass-nya. Dengan inheritance, class yang baru (subclass) akan mirip dengan class yang lama (superclass) namun memiliki karakteristik yang baru.

Dalam Java, subclass hanya bisa memiliki satu superclass (*single inheritance*) sedangkan superclass bisa memiliki satu subclass atau lebih. Untuk menerapkan inheritance, gunakan statement “*extends*”.

Keyword “*super*” digunakan oleh subclass untuk memanggil constructor, atribut dan method yang ada pada superclass-nya.

Contoh untuk memanggil constructor milik superclass-nya :

super() super(parameter)

Contoh untuk memanggil atribut dan method milik superclass-nya :

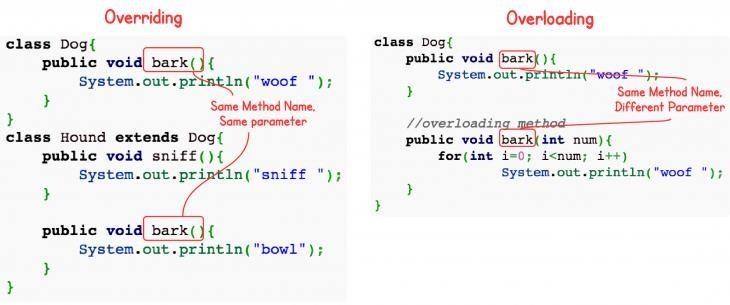
super.namaAtribut

super.namaMethod(parameter)

# Overriding vs Overloading

**Overloading** adalah diperbolehkannya dalam sebuah class memiliki lebih dari satu nama function/method yang sama tetapi memiliki parameter/argument yang berbeda.

Sedangkan **Overriding** adalah kemampuan dari subclass untuk memodifikasi method dari superclass-nya, yaitu dengan cara menumpuk (mendefinisikan kembali) method superclass-nya.

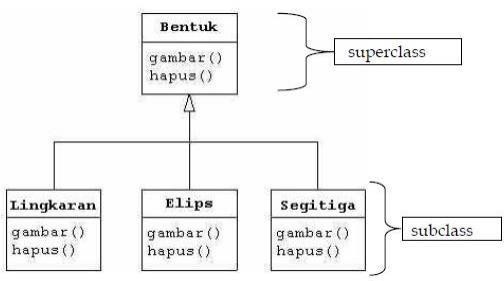


Gambar 5.1 Perbedaan Overriding dan Overloading

# Polimorfisme

Polimorfisme mempunyai makna sesuatu yang memiliki banyak bentuk, yaitu memiliki nama sama, tetapi memiliki kelakuan (behaviour) yang berbeda. Teknik polimorfisme digunakan untuk menyatakan satu nama yang merujuk (reference) ke beberapa fungsi yang berbeda.

Java menginterprestasikan polimorfisme dengan 3 cara yaitu method overloading, pewarisan, method overriding dengan interface java.



Gambar 5.2 UML Class Bentuk Perhatikan gambar diagram di atas !

Class Bentuk yang merupakan class induk (superclass) dari class Lingkaran, Elips dan Segitiga mempunyai method gambar() dan hapus(). Class-class anak (subclass) juga mempunyai method gambar() dan hapus(). Meskipun keempat class tersebut mempunyai nama method yang sama, tetapi isi (source code/yang dilakukan/output) dari masingmasing method tersebut berbeda.

Jika kita menginginkan sebuah objek yang dapat memanggil setiap method (yaitu method gambar & hapus) yang ada pada setiap class (pada superclass maupun subclass), maka gunakanlah teknik Polimorfisme. Polimorfisme hanya berlaku pada method dan tidak berlaku untuk atribut. Untuk mendapatkan operasi Polimorfisme dari suatu method, maka method tersebut haruslah merupakan method yang ada di class induk (lihat diagram diatas bahwa method gambar() dan hapus(), selain terdapat

di class-class turunan class Bentuk, juga terdapat di class Bentuk).

Perhatikan bahwa Polimorfisme tidak sama dengan overloading !!!

# Interface

Secara sederhana, Object Interface adalah sebuah ‘kontrak’ atau perjanjian implementasi method. Bagi class yang menggunakan object interface, class tersebut harus mengimplementasikan ulang seluruh method yang ada di dalam interface. Dalam pemrograman objek, penyebutan object interface sering disingkat dengan ‘Interface’ saja.

Contoh : Jika terdapat class komputer, interface bisa dicontohkan dengan ‘mouse’. Di dalam interface mouse, kita bisa membuat method seperti klik\_kiri(), klik\_kanan(), dan double\_klik(). Jika class laptop ‘menggunakan’ interface mouse, maka class tersebut harus membuat ulang method klik\_kiri(), klik\_kanan(), dan double\_klik().

Untuk mendeklarasikan sebuah interface gunakan sintaks :

interface namaInterface{

//isi dari interface }

Cara menggunakan suatu interface adalah dengan mengimplementasikan interface tersebut pada class yang menggunakannya. Selain itu, anda juga harus mendefinisikan secara detail method-method yang ada pada interface tersebut.

public class NamaClass implements namaInterface{ //method dan isi method pada namaInterface

}

Interface lebih berperan untuk menyeragamkan method. Ia tidak masuk kedalam struktur class seperti abstract class.

Interface Mouse:

public interface Mouse{

String jenis = “Laser Mouse”;

//Method Interface harus public public void klik\_kanan(); public void klik\_kiri();

//Tidak boleh ada method normal/biasa

//yang terdapat isi method-nya

}

Implementasi interface Mouse pada class Komputer :

public class Komputer implemenets Mouse{ public void klik\_kanan(){

System.out.println(“Mouse computer : “

+”klik Kanan… “);} public void klik\_kiri(){

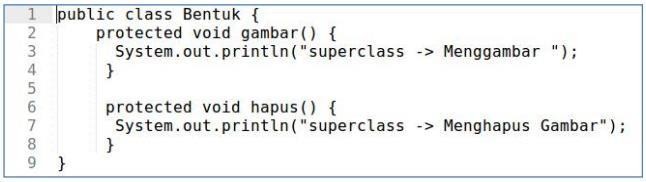
System.out.println(“Mouse computer : “

+”klik Kiri… “);}

}

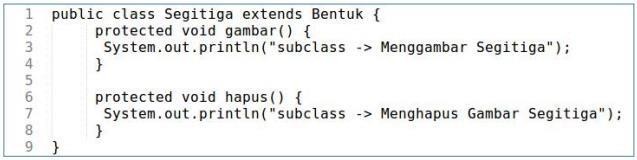
# PRAKTIKUM

* + 1. **Program Kelas Bentuk Bentuk.java**



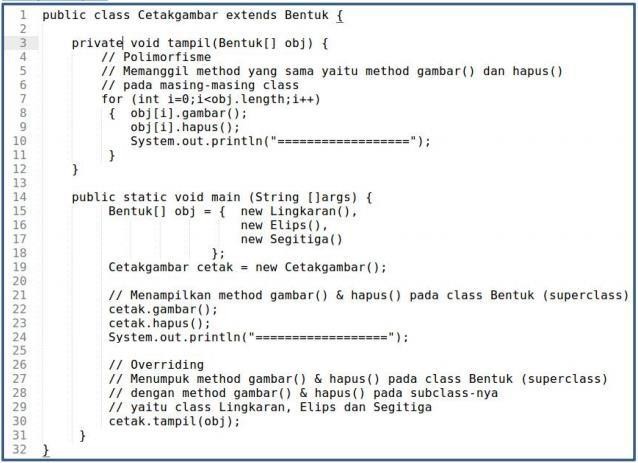
# Lingkaran.java

**Segitiga.java**



# Elips.java

**Cetakgambar.java**



# Program PoliKendaraan

**Kendaaran.java**

**package** kendaraan;

**class** Kendaraan { **protected int** roda; **protected** String nama,jenis;

**protected** Kendaraan(**int** rda,String nma,String jns){

roda=rda;

}

nama=nma;

jenis=jns;

**protected void** infoKendaraan(){ System.***out***.println("~~~"); System.***out***.println("Nama : "+nama); System.***out***.println("Jenis : "+jenis); System.***out***.println("Roda : "+roda);

}

}

**class** mobil **extends** Kendaraan{

**protected** mobil(**int** rda, String nma, String jns) {

**super**(rda, nma, jns);

} }

**class** motor **extends** Kendaraan{

**public** motor(**int** rda,String nma,String jns){

**super**(rda, nma, jns);

} }

**class** tank **extends** Kendaraan{ String keterangan;

**public** tank(**int** rda,String nma,String jns,String ktrg){

**super**(rda, nma, jns); keterangan=ktrg;

}

**public void** infoKendaraan(){

**super**.infoKendaraan(); System.***out***.println("Keterangan : "+keterangan);

} }

**class** pesawat **extends** Kendaraan{

**public** pesawat(**int** rda, String nma, String jns) {

**super**(rda, nma, jns);

} }

# PoliKendaraan.java

|  |
| --- |
| **package** kendaraan; **public class** PoliKendaraan {  **public static void** main(String[] args) { Kendaraan mbl = **new** mobil(4, "Ford", "Mobil");  Kendaraan mtr = **new** motor(2, "Ninja", "motor"); |
| Kendaraan pswt = **new** pesawat(10, "Garuda", "Pesawat"); Kendaraan tnk = **new** tank(0, "Leopard 2A7", "Tank", "Menggunakan rantai");  mbl.infoKendaraan(); mtr.infoKendaraan(); tnk.infoKendaraan(); pswt.infoKendaraan(); } |
| } |

Output program akan seperti berikut :

# TUGAS

1. Tugas sebelumnya meminta anda untuk membuat sebuah program Java dengan struktur OOP untuk suatu rumah sakit. Setelah anda menyelesaikan proyek tersebut rumah sakit datang kembali untuk meminta peningkatan ke program mereka!
2. Tambahkan kelas pewarisan dari kelas Pasien untuk, Pasien Rawat Inap, Pasien Poli, dan Pasien UGD (Pastikan setiap kelas memiliki perbedaan konstruktor dan metode yang sesuai dengan tujuan dari pewarisan tersebut).
3. Jangan lupa untuk mengimplementasikan polimorfisme pada setiap kelas tersebut (implementasi dibebaskan kepada anda).
4. Tambahkan 3 interface untuk program ini dimana dua diantaranya adalah mendaftarkan pasien ke rumah sakit, dan mengeluarkan pasien dari rumah sakit, interface ketiga dibebaskan kepada anda.
5. Simpan proyek Anda.