LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK



Laporan Untuk Melengkapi Tugas Terstruktur Mata Kuliah Pemrograman Berorientasi Object

Dosen Pengajar :

M. Imron Rosyadi, MM

Disusun oleh :

TEKNIK INFORMATIKA-3A

Yerico Armi Nanlohy ( 201869040023 )

**UNIVERSITAS YUDHARTAPASURUAN**

Jl.Yudharta No.7 (Pondok Pesantren Ngalah)

Sengonagung Purwosari Pasuruan Telp.(0343)611186

**2019**

**KATA PENGANTAR**

Dengan memanjatkan puji syukur kehadirat Tuhan Yang Maha Esa, atas segala limpahan rahmat dan karunia-Nya kepada kita semua, sehingga saya dapat menyelesaikan laporan praktikum ini.

Saya menyadari bahwa di dalam pembuatan laporan ini berkat bantuan dan tuntunan Tuhan Yang Maha Esa dan tidak lepas dari bantuan berbagai pihak, untuk itu dalam kesempatan ini saya menghaturkan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang membantu dalam pembuatan laporan ini.

Saya menyadari bahwa dalam proses penulisan laporan ini masih dari jauh dari kesempurnaan baik materi maupun cara penulisannya. Namun demikian, saya telah berupaya dengan segala kemampuan dan pengetahuan yang dimiliki sehingga dapat selesai dengan baik dan oleh karenanya, saya dengan rendah hati menerima masukan, saran dan usul guna penyempurnaan laporan ini.

Akhirnya saya berharap semoga makalah ini dapat bermanfaat bagi seluruh pembaca.

Pasuruan, 5 Desember 2020  
Penyusun,

Yerico Armi Nanlohy

**DAFTAR ISI**

**HALAMAN SAMPUL i**

**KATA PENGANTAR ii**

**DAFTAR ISI iii**

**BAB I. PENDAHULUAN 1**

1.1 Latar Belakang 1

1.2 Tujuan 2

1.3 Manfaat 2

**BAB II. PEMBAHAAN 3**

2.1Landasan Teori 3

2.1.1Java OOP 3

2.1.2 Java Class&Object 3

2.1.3 Java Class Attribute 4

2.1.4 Java Class Method 4

2.1.5 Java Constructor 5

2.1.6 Java Modifire 5

2.1.7 Java Encapsulation 6

2.1.8 Java API 6

2.1.9 Java Inherintance 6

2.1.10 Java Polymorphism 7

2.1.11 Java Iner Class 7

2.1.12 Java Interface 7

2.1.13 Java Abstraction 7

2.1.14 Java Enums 8

2.1.15 Java User Input 8

2.1.16 Java Date 8

2.1.17 Java Array List 8

2.1.18 Java Hash Map 9

2.1.19 Java Wraper Class 9

2.1.20 Java Exception 10

**BAB III. PRAKTIKUM 11**

3.1Praktikum 11

3.1.1 Java Constructor 11

3.1.2 Java Modifire 14

3.1.3 Java Encapsulation 16

3.1.4 Java API 19

3.1.5 Java Inherintance 22

3.1.6 Java Polymorphism 24

3.1.7 Java Iner Class 27

3.1.8 Java Interface 31

3.1.9 Java Abstraction 34

3.1.10 Java Enums 36

3.1.11 Java User Input 41

3.1.12 Java Date 45

3.1.13 Java Array List 47

3.1.14 Java Hash Map 49

3.1.15 Java Wraper Class 51

3.1.16 Java Exception 52

**BAB IV. PENUTUP 55**

4.1. Kesimpulan 55

4.2. Saran 55

**DAFTAR PUSTAKA 56**

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Pemrograman Berorientasi Objek (*Object Oriented Programming* atau OOP) merupakan paradigm pemrograman yang berorientasikan kepada objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untu merancang aplikasi dan program computer. Semua data dan fungsi di dalam paradigm ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Pada jaman sekarang, banyak bahasa pemrograman yang mendukung OOP.

OOP adalah paradigm pemrograman yang cukup mendominasi saat ini, karena mampu memberikan solusi kaidah pemrograman modern. Meskipun demikian, bukan berarti bahwa pemrogaman procedural sudah tidak layak lagi. OOP diciptakan karena dirasakan masih adanya keterbatasan pada bahasa pemrograman tradisional. Konsep OOP sendiri adalah semua pemecahan masalah dibagi kedalam objek. Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti *inheritance* atau pewarisan, *encapsulation* atau pembungkusan, dan *polymorphism* atau kebanyakrupaan. Konsep-konsep ini merupakan fundamental dalam orientasi objek yang perlu dipahami serta digunakan dengan baik, dan menghindari penggunaanya yang tidak tepat.

OOP lebih mudah dipelajari bagi pemula dibandingkan dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

* 1. **Tujuan**

1. Untuk melengkapi nilai tugas mata kuliah Pemrograman Berorientasi Objek.
2. Mampu membuat program java pada Netbeans berdasarkan aturan OOP.
3. Untuk mengkaji kembali pengetahuan yang telah didapatkan pada kuliah Pemrograman Berorientasi Objek.
   1. **Manfaat**
4. Dapat melengkapi nilai tugas mata kuliah Pemrograman Berorientasi Objek.
5. Dapat membuat program Java pada Netbeans berdasarkan ilmu aturan OOP.
6. Dapat mengkaji kembali pengetahuan yang telah didapatkan pada kuliah Pemrograman Berorientasi Objek.

**BAB II**

**PEMBAHASAN**

* 1. **Landasan Teori**
     1. Java OOP

OOP adalah singkatan dari **Object-Oriented Programming**. pemrograman prosedural adalah tentang menulis prosedur atau metode yang melakukan operasi pada data, sedangkan pemrograman berorientasi objek adalah tentang menciptakan benda-benda yang mengandung data dan metode. pemrograman berorientasi objek memiliki beberapa keunggulan dibandingkan pemrograman prosedural:

* OOP lebih cepat dan lebih mudah untuk mengeksekusi
* OOP menyediakan struktur yang jelas untuk program
* OOP membantu untuk menjaga kode Java DRY "Don’t Try Yourself", dan membuat kode lebih mudah untuk mempertahankan, memodifikasi dan debug
* OOP memungkinkan untuk membuat aplikasi yang dapat digunakan kembali penuh dengan kode kurang dan waktu pengembangan yang lebih pendek
  + 1. Java Class dan Object

Classmerupakan cetak biru (blue print) dari objek atau dengan kata lain sebuah class menggambarkan ciri – ciri objek secara umum.Sebagai contoh Honda Beat, Yamaha R6, dan Suzuki Satria merupakan objek dari class sepeda motor.

Kesatuan entitas (benda), baik yang berwujud nyata ataupun hanya suatu system / konsep yang memiliki sifat, karakter dan fungsi. Contoh :

|  |
| --- |
| publicclassMyClass{  int x =5;  publicstaticvoidmain(String[] args){  MyClass**myObj**=newMyClass();  System.out.println(myObj.x);  }} |

* + 1. Java Class Attributes

Segala sesuatu di Jawa dikaitkan dengan **Class** dan **Object**, bersama dengan **Attributes** dan **Method**. Sebagai contoh: dalam kehidupan nyata, mobil adalah obyek. Mobil memiliki **atribut** , seperti berat badan dan warna, dan **metode** , seperti drive dan rem. Contoh : Class MyClass dengan Attributes x dan y.

|  |
| --- |
| publicclassMyClass{  int x =5;  int y =3;  } |

* + 1. Java Class Method

Methodadalah kumpulan program yang mempunyai nama. Method merupakan sarana bagi programmer untuk memecah program menjadi bagian-bagian yang kecil agar jadi lebih kompleks sehingga dapat di gunakan berulang-ulang. Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object. Method didefinisikan pada class akan tetapi dipanggil melalui object. Contoh, pada object Mobil : terdapat method drive dan rem. Contoh :

|  |
| --- |
| publicclassMyClass{  staticvoidmyMethod(){  System.out.println("Hello World!");  }  publicstaticvoidmain(String[] args){  myMethod();  }  }  // Outputs "Hello World!" |

* + 1. Java Constructors

Construktoradalah method yang secara otomatis dipanggil/dijalankan pada saat sebuah class diinstransi. Jika dalam sebuah class tidak terdapat construktor maka secara otomatis java akan membuatkan sebuah default construktor. Nama construktor harus sama dengan nama class dan tidak boleh memiliki tipe return value. Sama halnya dengan method, construktor dapat memiliki satu atau banyak parameter maupun tanpa parameter.

* + 1. Java Modifiers

Access Modifier adalah sebuah “hak akses” yang diberikan kepada sebuah variabel/method/class dengan tujuan untuk menjaga integritas dari data tersebut ketika ingin diakses object lain. Hak akses sendiri diberikan oleh pembuat program sebagai salah satu bentuk implementasi dari OOP itu sendiri. Dengan adanya Access Modifier, kita dapat membatasi resource-resource mana saja yang dapat diakses oleh object tertentu, turunannya, ataupun oleh method tertentu. Di bahasa JAVA sendiri ada 4 jenis access modifier yang dapat digunakan :

1. Public

Access modifier public mempunyai hak akses paling luas dibanding yang lainnya. Karena aksesnya sangat luas, maka access modifier ini biasanya digunakan untuk method setter getter sesuai konsep OOP.

1. Prottected

Access modifier protected biasanya digunakan untuk mewariskan variabel yang ada di super class terhadap child class.

1. Non Access Modifire

Sesuai namanya, hak akses yang satu ini tidak perlu dituliskan di method/variabelnya. Dengan hak akses ini, variabel/method dapat diakses dari class lain asalkan masih dalam satu package yang sama.

1. Private

Access modifier private bersifat tertutup. Sesuai dengan konsep OOP Encapsulation, maka setiap variabel wajib untuk dilindungi hak aksesnya secara langsung dari luar. Oleh karena itu, variabel diberikan hak akses private dan untuk melakukan pengaksesan/perubahan data digunakan setter getter.

* + 1. Java Encapsulation

Enkapsulasi (encapsulation) adalah untuk memastikan bahwa data “sensitive” disembunyikan dari pengguna. Artinya encapsulation pada java digunakan untuk membungkus class dan menjaga apa saja yang ada dalam class baik itu method maupun attribute , agar tidak dapat di akses oleh kelas lain. Untuk mencapai ini maka perlu diperhatikan hal-hal berikut ini :

* Mendeklarasikan variabel / attribute kelas sebagai Private
* Memberikan metode public **Get** dan **Set**  untuk mengakses dan memperbarui nilai suatu Private variabel

**Get** dan **Set**

Variabel Private hanya dapat diakses didalam kelas yang sama (kelas luar tidak memiliki akses). Namun, memungkinkan mengaksesnya jika diberi tambahan metode Public **Get** dan **Set.**

Get adalah metode mendapatkan nilai variabel , dan Set metode mengatur nilaiSintaks untuk keduanya adalah bawa dimulai dengan salah satu get atau set , diikuti oleh nama variabel , dengan huruf pertama menggunakan huruf kapital.

* + 1. JavaAPI

API Java adalah perpustakaan kelas prewritten, yang bebas untuk digunakan, termasuk dalam Java Development Environment.Dalam library API berisi komponen untuk mengelola input, pemrograman database, dan lebih banyak lagi.

Library ini dibagi menjadi **paket** dan **kelas** . Berarti Anda baik dapat mengimpor satu kelas (bersama dengan metode dan atribut), atau seluruh paket yang berisi semua kelas yang milik paket tertentu.

* + 1. Java Inheritance

Konseppewarisan (inheritance) : Proses pembentukan kelas baru dari kelas yang sudah ada menjadi kelas Superclass dan kelas Subclasses. Proses pembentukan kelas menghasilkan hierarki kelas yaitu puncak hierarki yang disebut **kelas abstrak**.

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.

* + 1. Java Polymorphism

Polymorphism merupakan konsep sederhana dalam bahasa pemrograman berorientasi obyek yang berarti kemampuan sebuah obyek untuk menghasilkan aksi yang berbeda, bila method yang sama dipanggil, dimana aksi method yang dikerjakan tergantung dari tipe obyeknya.

Polymorphism merupakan suatu konsep yang menyatakan sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku berbeda. Jadi dapat disimpulkan bahwa polymorphism adalah satu objek yang memiliki banyak bentuk.

* + 1. Java Inner Class

Inner class adalah kelas yang disisipkan didalam kelas lain. Fungsi kelas sisipan ini adalah mendukung suatu proses yang dijalankan  oleh kelas utamanya. Inner class bersifat tersarang terhadap kelas-kelas utamnya.

* + 1. Java Interface

Interface adalah sekumpulan konstanta dana tau deklarasi mothod tanpa munyertakan / menuliskan body methodnya. Interface bisa digunakan untuk mendeklarasikan koleksi method dan konstanta yang dapat digunakan oleh satu atau lebih class.

* + 1. Java Abstaction

A**bstraction** adalah proses menyembunyikan rincian tertentu dan hanya menampilkan informasi penting kepada pengguna.

Abstraksi dapat dicapai dengan baik **kelas abstrak atau inerface.** kata kunci Abtraksi adalah pengubah non-akses, digunakan untuk kelas dan metode :

* **Kelas abstrak:** adalah kelas terbatas yang tidak dapat digunakan untuk membuat objek (untuk mengaksesnya, harus diwariskan dari kelas lain).
* **Metode abstrak:** hanya dapat digunakan di kelas abstrak, dan tidak memiliki tubuh. Tubuh disediakan oleh subclass (diwarisi dari).
  + 1. Java Enums

Enum (enumeration)adalah salah satu kelas library di J2SE (versi 5), yang dapat kita gunakan untuk membuat kumpulan variable yang telah didefinisikan menjadi tipe data **konstanta** (variabel tidak bisa diubah, seperti final variabel).setiap objek baru yang didefinisikan berdasarkan tipe data Enum, objek tersebut dapat mengakses atribut atau method (Enum), lalu mengaplikasikan nilai atau value yang ada di dalamnya. Enum bisa juga disebut array, namun dalam bentuk konstanta, dan juga bertipe class.

* + 1. Java User Input

Kelas **Scanner** digunakan untuk mendapatkan input pengguna, dan ditemukan dalam paket java.util . Untuk menggunakan kelas **Scanner ,** buat objek kelas dan gunakan salah satu metode yang tersedia yang ditemukan dalam dokumentasi kelas **Scanner** .

* + 1. Java Date

Java tidak memiliki kelas tanggal bawaan , tetapi dapat mengimpor paket java.time agar berfungsi dengan API tanggal dan waktu. Paket terdiri atas beberapa kelas tanggal dan waktu .Sebagai contoh :

|  |  |
| --- | --- |
| Class | Deskripsi |
| Local Date | Menjelaskan tentang tanggal (tahun,bulan,hari  (yyyy-MM-dd)) |
| Local Time | Menjelaskan tentang waktu (jam,menit,detik dan seperinu detik (HH-mm-ss-zzz)) |
| Local Date Time | Menjelaskan tentang tanggal dan waktu (yyyy-MM-dd-HH-mm-ss.zzz) |
| Date Time Formatter | Menampilkan dan menguraikan objek tanggal dan waktu |

* + 1. Java Arraylist

Kelas ArrayList adalah [array](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.w3schools.com/java/java_arrays.asp&xid=25657,15700021,15700186,15700191,15700256,15700259,15700262,15700265,15700271,15700283&usg=ALkJrhiTH6rPlA-QuvlB_z7aByHZNOCRkQ) resizable, yang dapat ditemukan dalam paket java.util.Perbedaan antara array bawaan dan ArrayList di Java, adalah bahwa ukuran array tidak dapat dimodifikasi (jika Anda ingin menambah atau menghapus elemen ke / dari array, Anda harus membuat yang baru). Sementara elemen dapat ditambahkan dan dihapus dari ArrayList kapan pun Anda mau.

* + 1. Java HashMap

Anda pelajari dari bab sebelumnya, bahwa Array menyimpan barang-barang sebagai koleksi memerintahkan, dan Anda memiliki akses mereka dengan nomor indeks ( inttipe). Sebuah HashMapNamun, toko item dalam " **key** / **value** " pasangan, dan Anda dapat mengaksesnya dengan indeks jenis lain (misalnya String). HashMap mempunai banyak metode :

1. Add Item : Untuk menambah item, menggunakan metode put().
2. Access an Item : Untuk mengakses nilai item, menggunakan metode get().
3. Remove an Item : Untuk menghapus item, mengunakan metode remove(). Untuk menghapus semua item menggunakan metode clear().
4. HashMap Size : Unuk mengetahui berapa banyak item yang ada, menggunakan metode size.
5. Loop Through a HashMap : Gunakan keyset() Jika hanya ingin key, dan gunakanvalues() jika hanya inin value.
6. Other Type : Kunci dan nilai-nilai dalam HashMap sebenarnya objek. Dalam contoh di atas, kami menggunakan objek tipe "String". Ingat bahwa String di Jawa adalah obyek (bukan tipe primitif). Untuk menggunakan jenis lain, seperti int, Anda harus menentukan setara [kelas wrapper](https://www.w3schools.com/java/java_wrapper_classes.asp) : Integer. Untuk jenis primitif, gunakan: Booleanuntuk boolean, Characteruntuk char,  Doubleuntuk ganda, dll.
   * 1. JavaWrapper **Class**

Di dalam bahasa pemrograman Java, *Wrapper Class* (Kelas Pembungkus) adalah suatu mekanisme yang digunakan untuk mengconvert atau mengubah suatu nilai yang di definisikan, dari tipe data **primitif**menjadi sebuah nilai dengan tipe data **referensi**(Objek). Selain itu, Wrapper Class mendukung method dari tipe data primitif, contohnya seperti Boolean, Character, Integer, Long, Float, dsb.

Pada bahasa pemrograman Java, terdapat 2 jenis tipe data, yaitu primitif dan referensi (objek), Tipe Data *Primitif*adalah tipe data yang tidak memiliki method, hanya memiliki data saja. Tipe data ini bukanlah sebuah object, sedangkan Tipe Data *Referensi*adalah tipe data yang digunakan untuk memegang referensi dari sebuah object ( instance dari class).Dan berikut ini merupakan table yang menjelaskan macam-macam tipe data primitif dan referensi pada Java:

|  |  |
| --- | --- |
| **Tipe Data Primitif** | **Tipe Data Referensi** |
| char | Character |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| boolean | Boolean |
| float | Float |
| double | Double |

metode berikut ini digunakan untuk mendapatkan nilai yang terkait dengan objek pembungkus yang sesuai:

intValue() , byteValue() , shortValue() , longValue() , floatValue() , doubleValue() , charValue() , booleanValue() .

* + 1. Java Exeption

Ketika mengeksekusi kode Java, kesalahan yang berbeda dapat terjadi: coding kesalahan yang dibuat oleh programmer, kesalahan karena masukan yang salah, atau hal-hal tidak terduga lainnya.

Ketika terjadi kesalahan, Jawa biasanya akan berhenti dan menghasilkan pesan kesalahan. Istilah teknis untuk ini adalah: Java akan melemparkan sebuah **pengecualian** (melempar kesalahan).

**BAB III**

**PRAKTIKUM**

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Constructor

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

public class Constructor {

long modelnim;

String modelname;

String modelalamatrumah;

String modeljurusan;

int modelangkatan;

public Constructor() {

modelnim = 201869040023L;

modelname = "Yerico Armi Nanlohy";

modelalamatrumah = "jl.Raya Sukorejo";

modeljurusan = "Teknik Informatika";

modelangkatan = 2018;

}

public static void main(String[] args) {

Constructor profil = new Constructor();

System.out.println("nim = "+profil.modelnim);

System.out.println("name = "+profil.modelname);

System.out.println("alamat rumah = "+profil.modelalamatrumah);

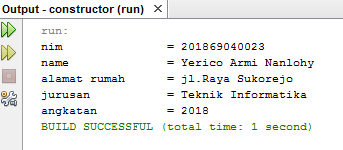
System.out.println("jurusan = "+profil.modeljurusan);

System.out.println("angkatan = "+profil.modelangkatan);

}

}

**Ouput Program :**

****

**Penjelasan Program :** Program ini digunakan untuk menampilkan data berupa biodata mahasiswa. Program ini memiliki class bernama constructor dan method bernama main.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Modifire (public)

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

public class Public {

public String fname = "Yerico Armi Nanlohy";

public String lname = "yeri";

public String email = "yericonanlohy@gmail.com";

public int age = 20;

public static void main(String[] args) {

Public myObj = new Public();

System.out.println("Name : " + myObj.fname + " " + myObj.lname);

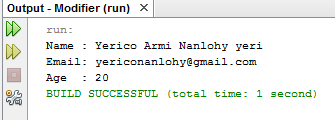
System.out.println("Email: " + myObj.email);

System.out.println("Age : " + myObj.age);

}

}

**Ouput Program :**

****

**Penjelasan Program :** Modifier public akan membuat member dan class bisa di akses dari mana saja. Program ini digunakan untuk menampilkan data seseorang.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java API

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

public class Profil {

public static void main(String[] args){

BufferedReader input = new BufferedReader (new InputStreamReader (System.in));

String nama;

String alamat;

String jurusan;

String jenisKelamin;

try{

System.out.print ("nama :");

nama = input.readLine ();

System.out.print ("alamat :");

alamat = input.readLine ();

System.out.print ("jurusan :");

jurusan = input.readLine ();

System.out.print ("jenisKelamin :");

jenisKelamin = input.readLine ();

System.out.println("nama "+ nama);

System.out.println("alamat "+ alamat);

System.out.println("jurusan "+ jurusan);

System.out.println("jenisKelamin "+ jenisKelamin);

}catch (IOException ex){

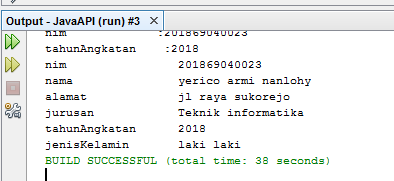
System.out.println("Terjadi kesalahan pada input");

}

}

}

**Output Program :**

****

**Penjelasan Program :** Program ini digunakan untuk mengisi data berupa biodata mahasiswa. Program ini memiliki class bernama biodata dan method bernama main.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Inheritance

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

class Vehicle {

protected String brand = "Ford"; // Vehicle attribute

public void honk() { // Vehicle method

System.out.println("Tuut, tuut!");

}

}

class Car extends Vehicle {

private String modelName = "Mustang"; // Car attribute

public static void main(String[] args) {

// Create a myCar object

Car myCar = new Car();

// Call the honk() method (from the Vehicle class) on the myCar object

myCar.honk();

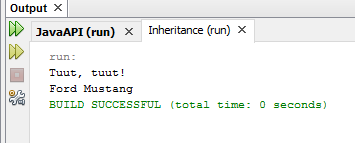
// Display the value of the brand attribute (from the Vehicle class) and the value of the modelName from the Car class

System.out.println(myCar.brand + " " + myCar.modelName);

}

}

**Output Program :**

****

**Penjelasan Program :** Program ini digunakan untuk mengisi data menghitung luas serta keliling persegi. Program ini memiliki class bernama persegi dan method tipe void.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Polymorphism

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

public class Objects {

public void ObjectsSound() {

System.out.println("The Objects sound");

System.out.println("=================");

}

}

class Bell extends Objects {

public void ObjectsSound() {

System.out.println("Suaranya Bell : Kriiiinngggg");

}

}

class Angin extends Objects {

public void ObjectsSound() {

System.out.println("Suaranya Kipas Angin : Wusssshhh");

}

}

class Telepon extends Objects {

public void ObjectsSound() {

System.out.println("Suaranya Telepon : Triingg triing triing");

}

}

class MyMainClass {

public static void main(String[] args) {

Objects myObjects = new Objects();

Objects myBell = new Bell();

Objects myAngin = new Angin();

Objects myTelepon = new Telepon();

myObjects.ObjectsSound();

myBell.ObjectsSound();

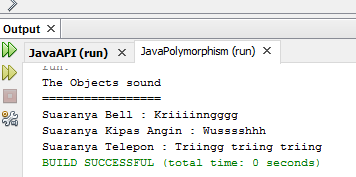
myAngin.ObjectsSound();

myTelepon.ObjectsSound();

}

}

**Output Program :**



**Penjelasan Program :** Program ini digunakan untuk menampilkan suara benda-benda. Program ini memiliki class bernama Objects dan method tipe void.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Inner Class

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

public class KelasLuar {

private class Baju{

private String merk = "Supreme";

private float harga = 360.0f;

private void jalankan(){

System.out.println("Merk Outfit: "+merk);

System.out.println("Harga Outfit: "+harga);

}

}

private class Pengguna{

private String nama = "Yerico";

private int umur = 20;

private void identitas(){

System.out.println("Nama Saya: "+nama);

System.out.println("Usia Saya: "+umur);

}

}

public static void main(String[] args){

KelasLuar outerclass = new KelasLuar();

KelasLuar.Baju data1 = outerclass.new Baju();

KelasLuar.Pengguna data2 = outerclass.new Pengguna();

System.out.println("===== DATA DARI CLASS MOBIL ========");

data1.jalankan();

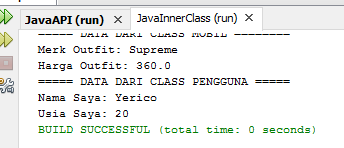
System.out.println("===== DATA DARI CLASS PENGGUNA =====");

data2.identitas();

}

}

**Output Program :**



**Penjelasan Program :**Program ini digunakan untuk menampilkan data dari pemesanan baju. Program ini memiliki class bernama KelasLuar dan memiliki method main.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Interface

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

interface SuaraHewan {

public void SuaraHewan(); // interface method (does not have a body)

public void Awake(); // interface method (does not have a body)

}

class Dog implements SuaraHewan {

public void SuaraHewan() {

System.out.println("Anjing Kalo Bangun: Guugk guuk guugk");

}

public void Awake() {

System.out.println("Gggrrrr!!");

}

}

class MyMainClass {

public static void main(String[] args) {

Dog myDog = new Dog();

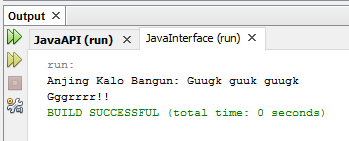
myDog.SuaraHewan();

myDog.Awake();

}

}

**Output Program :**



**Penjelasan Program :** Program ini menampilkan suara laki-laki, yaitu yuhuu saat berkata kemudian kkrr kkrr saat tidur. Program ini memiliki class bernama Humans dan memiliki method main.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Abstraction

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

abstract class Bebek {

public abstract void BebekSound();

public void sleep() {

System.out.println("kwak kwak kwak");

}

}

class Dog extends Bebek {

public void BebekSound() {

System.out.println("The bebek says: kwaak kwaak");

}

}

abstract class Bebekku {

public static void main(String[] args) {

Dog myDog = new Dog();

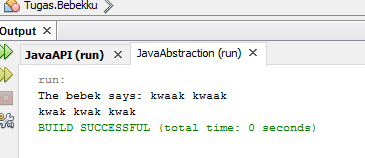
myDog.BebekSound();

myDog.sleep();

}

}

**Output Program :**

****

**Penjelasan Program :** Program ini menampilkan suara Pig, yaitu “wee wee” saat berkata kemudian “Zzzz” saat tidur. Program ini memiliki class bernama Animals dan memiliki method main.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Enums

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script Program :**

enum Day {

Newyear,

iedmubarak,

MerryChristmast

}

public class SwitchStatement {

public static void main(String[] args) {

Day myVar = Day.iedmubarak;

switch(myVar) {

case Newyear:

System.out.println("Happy New year");

break;

case iedmubarak:

System.out.println("Happy Ied Mubarak");

break;

case MerryChristmast:

System.out.println("Merry Christmast");

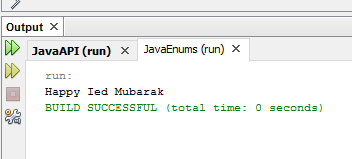
break;

}

}

}

**Output Program :**

****

**Penjelasan Program :** Program diatas ini melakukanPencetakan nama hari sesuai perintah dari pengguna seperti contoh output diatas yaitu “Valentine Days”.

**Latihan Praktikum**

**Pemrograman JAVA**

**Nama Program :** Java Exception

**Bahasa Pemrograman :** JAVA

**Compiler :** IDE NetBeans 8.2

**Script :**

public class Exception {

public static void main(String[]args){

int pembilang = 2;

int penyebut = 0;

int hasil = pembilang / penyebut;

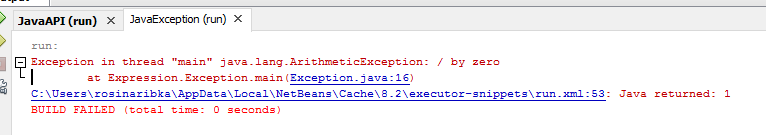
System.out.println("Hasil = " + hasil);

System.out.println("Statement Setelah blok try-catch");

}

}

**Output :**

****

**BAB IV**

**PENUTUP**

* 1. **KESIMPULAN**

Pemrograman Berorientasi Objek (*Object Oriented Programming* atau OOP) merupakanparadigma pemrograman yang berorientasikan kepada objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untu merancang aplikasi dan program computer. Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti *inheritance* atau pewarisan, *encapsulation* atau pembungkusan, dan *polymorphism* atau lainnya. Konsep-konsep ini merupakan fundamental dalam orientasi objek yang perlu dipahami serta digunakan dengan baik, dan menghindari penggunaanya yang tidak tepat. Dalam laporan praktikum ini dilengkapi dengan studi kasus serta hasil percobaan pada setiap babnya.

* 1. **SARAN**

Laporan praktikum ini dapat dikembangkan lebih lanjut sehingga memajukan pendidikan teknologi informasi dan komunikasi Indonesia, terutama pada bidang studi pemrograman brorientasi objek menggunakan bahasa pemrograman java.

**DAFTAR PUSTAKA**

W3Schools. 2019. *Java*.

<https://www.w3schools.com/java/default.asp>

(diakses tanggal 24 Desember 2019 pukul 10.09 WIB)