LAPORAN PRAKTIKUM

BAHASA PEMOGRAMAN JAVA (OOP)



Nama dosen :

M.Imron Rosyadi S.Kom M.Kom

Nama :

Lailatul Afidah(201869040018)

FAKULTAS TEKNIK

PROGRAM STUDY TEKNIK INFORMATIKA

UNIVERSITAS YUDHARTA

2019

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan kesehatan, kekuatan, kesempatan serta limpahan rahmat dan karunianya sehingga penulis dapat menyelesaikan laporan praktikum pada mata kuliah Bahasa Pemrograman Java (OOP) ini dengan baik. Penulisan laporan ini dibuat agar dapat memenuhi sebagi salah satu syarat untuk mendapatkan nilai tambahan serta dalam mengikuti ujian semester. Dengan sepenuh hati penulis menyadari bahwa tersusunnya tugas ini berkat bantuan dan bimbingan dari semua pihak baik berupa material, spiritual, maupun informasi. Oleh karena itu dalam kesempatan ini penulis tidak lupa mengucapkan terima kasih sebanyak- banyaknya kepada :

1. Kepada Tuhan Yang Maha Esa yaitu Allah Swt, karena dengan anugerahnya   
penulis dapat menyelesaikan laporan praktikum ini dengan baik.

2. Orangtua saya yang telah memberikan semangat dan dukungan sehingga laporan   
praktikum ini dapat terselesaikan dengan baik.

3. Bapak Imron Rosadi S.Kom, selaku dosen Bahasa Pemrograman Java (OOP).

4. Semua pihak yang telah membantu dalam penyusunan laporan ini.

Akhirnya penulis mohon maaf apabila ada kekurangan atau kesalahan dalam penyusunan laporan praktikum ini. Semoga laporan praktikum ini dapat bermanfaat bagi penulis, maupun pembacanya

Pasuruan, 12 Desember 2019

Penulis

Lailatul Afidah

DAFTAR ISI

KATA PENGANTAR

DAFTAR ISI

**BAB I PENDAHULUAN**

* 1. Latar Belakang 1
  2. Rumusan Masalah 1
  3. Tujuan penelitian 2

**BAB II LANDASA TEORI**

* 1. Konsep OOP 3
  2. Pengenalan object dan class 4
  3. Java class Attribute 5-7
  4. Java Class Mtehods 8-12
  5. Java Constructor 13-15
  6. Java Modifiers 16-23
  7. Java Encapsulation 24-25
  8. Java Packages/API 25-28
  9. Java Inheritence 28-30
  10. Java Polymorishm 30-32
  11. Java Inner Classes 33-36
  12. Java Abstraction 36-38
  13. Java Interface 38-41
  14. Java Enums 41-42
  15. Java User Input 43-44
  16. Java Date 44-50
  17. Java Array List 51-59
  18. Java Hash Map 60-62
  19. Java Wrapper 63-65
  20. Java Exception 65-66

**BAB III HASIL PERCOBAAN/PRAKTIKUM**

* 1. Hasil Percobaan 1 66-68
  2. Hasil Percobaan 2 68-70
  3. Hasil Percobaan 3 71
  4. Hasil Percobaan 4 72-75
  5. Hasil Percobann 5 76-79
  6. Hasil Percobaan 6 80
  7. Hasil Percobaan 7 81-82
  8. Hasil Percobaan 8 83
  9. Hasil Percobaan 9 84-85
  10. Hasil Percobaan 10 86
  11. Hasil Percobaan 11 87
  12. Hasil Percobaan 12 88
  13. Hasil Percobaan 13 88-89
  14. Hasil Percobaan 14 89
  15. Hasil Percobaan 15 90

**BAB IV ANALISIS HASIL PERCOBAAN**

* 1. Analisis Hasil Percobaan 1 91-92
  2. Analisis Hasil Percobaan 2 93-94
  3. Analisis Hasil Percobaan 3 94
  4. Analsis Hasil Percobaan 4 94-96
  5. Analisis Hasil Percobaan 5 96-99
  6. Analisis Hasil Percobaan 6 99-100
  7. Analisis Hasil Percobaan 7 100-101
  8. Analisis Hasil Percobaan 8 101
  9. Analisis Hasil Percobaan 9 102
  10. Analisis Hasil Percobaan 10 103
  11. Analisis Hasil Percobaan 11 103-104
  12. Analisis Hasil Percobaan 12 104-105
  13. Analisis Hasil Percobaan 13 106
  14. Analisis Hasil Percobaan 14 107
  15. Analisis Hasil Percobaan 15 107-108

**BAB V : PENUTUP**

**KESMIPULAN** 109

**SARAN** 109

**DAFTAR PUSTAKA** …………………………………………………………110

**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. **Rumusan Masalah**
2. Bagaimana pengertian OOP
3. Apa saja jenis-jenis yang ada dalam OOP
4. Bagaimana cara menjalankan program yang ada dalam OOP
5. **Tujuan Praktikum**
6. mengetahui pengertian OOP
7. Untuk mengetahui jeni-jenis OOP
8. Untuk megetahui cara menjalankan program pada OOP

**BAB II**

**LANDASAN TEORI**

* 1. **Konsep OOP**

OOP adalah singkatan **dari Pemrograman Berorientasi Objek** .

Pemrograman prosedural adalah tentang prosedur atau metode penulisan yang melakukan operasi pada data, sedangkan pemrograman berorientasi objek adalah tentang membuat objek yang berisi data dan metode.

Pemrograman berorientasi objek memiliki beberapa keunggulan dibandingkan pemrograman prosedural:

* OOP lebih cepat dan lebih mudah untuk dieksekusi.
* OOP menyediakan struktur yang jelas untuk program-program tersebut.
* OOP membantu menjaga kode Java KERING "Don't Repeat Yourself", dan membuat kode lebih mudah untuk mempertahankan, memodifikasi dan men-debug.
* OOP memungkinkan untuk membuat aplikasi yang dapat digunakan kembali secara penuh dengan kode yang lebih sedikit dan waktu pengembangan yang lebih singkat.

Kelas dan objek adalah dua aspek utama dari pemograman berorientasi objek. Lihatlah ilustrasi berikut untuk melihat perbedaan antara kelas dan objek:

|  |  |
| --- | --- |
| KELAS | OBJEK |
| Buah | Apel |
| Mangga |
|  | Pisang |

Jadi, kelas adalah template untuk objek, dan objek adalah turunan dari kelas. Ketika objek individu dibuat, mereka mewarisi semua variabel dan metode dari kelas.

**2.2 Pengenalan Objek dan Class**

Segala sesuatu di Java dikaitkan dengan kelas dan objek, bersama dengan attribut dan methodnya. Sebagai contoh: dalam kehidupan nyata, mobil adalah objek. Mobil memiliki **attribut** , seperti berat dan warna, dan **method** , seperti drive dan rem.

Kelas seperti konstruktor objek, atau "cetak biru" untuk membuat objek.

* Mendeklarsaikan Kelas

Sebuah kelas dideklarasikan dengan menggunakan kata keyword ***class***. kelas ini merupakan objek template yang berisi spesifikasi umum dari suatu objek

Membuat kelas :

public class MyClass {

int x = 5;

}

* Mendeklarasikan objek

Di Java, objek dibuat dari kelas. Kami telah membuat kelas bernama MyClass , jadi sekarang kita bisa menggunakan ini untuk membuat objek.

Untuk membuat objek MyClass , tentukan nama kelas, diikuti oleh nama objek, dan gunakan keyword **new**

public class MyClass {

int x = 5;

public static void main(String[] args) {

MyClass **myObj** = new MyClass();

System.out.println(myObj.x);

}

}

Output:

5

* 1. **Java Class Attribute**

Attributes merupakan nilai data  yang  terdapat pada  suatu object  yang berasal dari class. Attributes merepresentasikan karakteristik dari suatu object. Atau Atribut adalah sesuatu yang dimiliki oleh objek.  Setiap objek yang dibuat dari kelas yang sama akan memiliki atribut yang sama.

istilah "variabel" untuk x dalam contoh (seperti yang ditunjukkan di bawah). Ini sebenarnya adalah **atribut** dari kelas. Atau Anda bisa mengatakan bahwa atribut kelas adalah variabel dalam kelas:

public class MyClass {

int x = 5;

int y = 3;

}

* Mengakses Attribute

Anda bisa mengakses atribut dengan membuat objek kelas, dan dengan menggunakan sintaks dot ( . ):

Contoh berikut akan membuat objek kelas MyClass , dengan nama myObj . Kami menggunakan atribut x pada objek untuk mencetak nilainya:

public class MyClass {

int x = 5;

public static void main(String[] args) {

MyClass myObj = new MyClass();

System.out.println(myObj.x);

}

}

Outputan:

5

* Mengubah Nilai Attribute
* Menetapkan nilai attribute

Example kita mengubah nilai x menjadi 40

public class MyClass {

int x;

public static void main(String[] args) {

MyClass myObj = new MyClass();

myObj.x = 40;

System.out.println(myObj.x);

}

}

Output:

40

* Mengubah nilai yang ada menjadi nilai baru

Example mengubah nilai 10 menjadi 25

public class MyClass {

int x = 10;

public static void main(String[] args) {

MyClass myObj = new MyClass();

myObj.x = 25; // x is now 25

System.out.println(myObj.x);

}

}

Output:

25

* Jika ingin membuat banyak objek dalam satu kelas maka cukup mengubah nilai attribute di sabtu objek ,tanpa mengubah nilai attribute di yang lain :

Example

public class MyClass {

int x = 5;

public static void main(String[] args) {

MyClass myObj1 = new MyClass(); // Object 1

MyClass myObj2 = new MyClass(); // Object 2

myObj2.x = 25;

System.out.println(myObj1.x); // Outputs 5

System.out.println(myObj2.x); // Outputs 25

}

}

Output :

5

* Banyak Attribute

Anda juga bisa menentukan banyak attribute sebanyak yang anda inginkan

public class Person {

String fname = "John";

String lname = "Doe";

int age = 24;

public static void main(String[] args) {

Person myObj = new Person();

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

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

}

}

Output :

Jhon Doe

24

* 1. **Java Class Methods**

Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object. Method didefinisikan pada class akan tetapi dipanggil melalui object. Method menentukan prilaku objeck, akni apa yang terjadi ketika objeck itu di buat serta berbagai operasi yang dapat dilakukan objrk sepanjang hidupnya.

* Mengakses Methods

metode dideklarasikan dalam kelas, dan digunakan untuk melakukan tindakan tertentu:

Example :

Buatlah methods bernama myMethod() di Myclass:

public class MyClass {

static void myMethod() {

System.out.println("Hello World!");

}

}

**myMethod()** mencetak teks (aksi) ketika **di panggil** . Untuk memanggil metode, tulis nama metode diikuti oleh dua tanda kurung **()** dan tanda titik koma **; (***lihat warna kuning)*

public class MyClass {

static void myMethod() {

System.out.println("Hello World!");

}

public static void main(String[] args) {

myMethod();

}

}

// Outputs

"Hello World!"

* Method di bedakan menjadi 2 yaitu :
* Static

Staic yaitu dapat diakses tanpa membuat objek kelas.

* Non Static

Non Static yaitu hanya dapat diakses oleh objek itu sendiri.

Dalam contoh di atas, kami membuat metode static , yang berarti dapat diakses tanpa membuat objek kelas, tidak seperti public , yang hanya dapat diakses oleh objek:

## Metode Akses dengan Objeck

Contoh :

Buatlah method dengan kelas bernama **myCar** . Panggil method **fullThrottle();** dan **speed();** pada objek myCar

public class MyClass {

// Static method

static void myStaticMethod() {

System.out.println("Static methods can be called without creating objects");

}

// Public method

public void myPublicMethod() {

System.out.println("Public methods must be called by creating objects");

}

// Main method

public static void main(String[] args) {

myStaticMethod(); // Call the static method

// myPublicMethod(); This would compile an error

MyClass myObj = new MyClass(); // Create an object of MyClass

myObj.myPublicMethod(); // Call the public method on the object

}

}

Output:

Static methods can be called without creating objects  
Public methods must be called by creating objects

***Contoh* Scoude Code :**

// Create a Car class

public class Car {

// Create a fullThrottle() method

public void fullThrottle() {

System.out.println("The car is going as fast as it can!");

}

// Create a speed() method and add a parameter

public void speed(int maxSpeed) {

System.out.println("Max speed is: " + maxSpeed);

}

// Inside main, call the methods on the myCar object

public static void main(String[] args) {

Car myCar = new Car(); // Create a myCar object

myCar.fullThrottle(); // Call the fullThrottle() method

myCar.speed(200); // Call the speed() method

}

}

Output:

The car is going as fast as it can!

Max speed is: 200

**Penjelasan dari contoh di atas :**

1) Kami menciptakan kelas Car khusus dengan kata kunci class

2) Kami menciptakan metode fullThrottle() dan speed() di kelas Car .

3) Metode fullThrottle() dan metode speed() akan mencetak beberapa teks, ketika mereka dipanggil.

4) Metode speed() menerima parameter int disebut maxSpeed - kita akan menggunakannya dalam 8) .

5) Untuk menggunakan kelas Car dan metodenya, kita perlu membuat objek dari Kelas Car .

6) Lalu, buka metode main() , yang Anda tahu sekarang adalah metode Java bawaan yang menjalankan program Anda (kode apa pun di dalam main dieksekusi).

7) Dengan menggunakan kata kunci new kami membuat objek Car dengan nama myCar .

8) Kemudian, kita memanggil metode fullThrottle() dan speed() pada objek myCar , dan menjalankan program menggunakan nama objek ( myCar ), diikuti oleh titik ( . ), Diikuti oleh nama metode ( fullThrottle(); dan speed(200); ). Perhatikan bahwa kita menambahkan parameter int 200 di dalam metode speed() .

* **Catatan Penting :**

1. Titik (.) Digunakan untuk mengakses atribut dan metode objek.
2. Untuk memanggil metode di Java, tulis nama metode diikuti dengan seperangkat tanda kurung () , diikuti dengan tanda titik koma ( ; ).
3. Kelas harus memiliki nama file yang cocok ( Car dan Car.java )

Untuk mengakses method dengan menggunakan objek kita bisa menggunakan banyak kelas. Dan yang harus di ingat nama file harus sesuai dengan nama kela

Dari contoh kelas Car di atas kita coba menggunakan banyak kelas :

#### **Car.java**

public class Car {

public void fullThrottle() {

System.out.println("The car is going as fast as it can!");

}

public void speed(int maxSpeed) {

System.out.println("Max speed is: " + maxSpeed);

}

}

#### **OtherClass.java**

class OtherClass {

public static void main(String[] args) {

Car myCar = new Car(); // Create a myCar object

myCar.fullThrottle(); // Call the fullThrottle() method

myCar.speed(200); // Call the speed() method

}

}

Maka Outputan nya akan sama:

The car is going as fast as it can!  
Max speed is: 200

* 1. **Java Constructors**

Konstruktor di Jawa adalah **metode khusus** yang digunakan untuk menginisialisasi objek. Konstruktor dipanggil ketika objek kelas dibuat. Ini dapat digunakan untuk mengatur nilai awal untuk atribut objek:

Contoh

Buat konstruktor:

// Create a MyClass class

public class MyClass {

int x; // Create a class attribute

// Create a **class constructor** for the MyClass class

public MyClass() {

x = 5; // Set the initial value for the class attribute x

}

public static void main(String[] args) {

MyClass myObj = new MyClass(); // Create an object of class MyClass (This will **call the constructor**)

System.out.println(myObj.x); // Print the value of x

}

}

// Outputs 5

Perhatikan bahwa nama konstruktor harus cocok dengan nama kelas , dan tidak boleh memiliki tipe kembali (seperti void ). **Perhatikan juga bahwa konstruktor dipanggil saat objek dibuat.**

Semua kelas memiliki konstruktor secara default: jika Anda tidak membuat konstruktor kelas sendiri, Java membuat satu untuk Anda. Namun, maka Anda tidak dapat menetapkan nilai awal untuk atribut objek.

* Constructor Parameters

Konstruktor juga dapat mengambil parameter, yang digunakan untuk menginisialisasi atribut.

Contoh berikut menambahkan parameter int y ke konstruktor. Di dalam konstruktor kita atur x ke y (x = y). Ketika kita memanggil konstruktor, kita meneruskan parameter ke konstruktor (5), yang akan menetapkan nilai x ke 5:

Contoh:

public class MyClass {

int x;

public MyClass(int y) {

x = y;

}

public static void main(String[] args) {

MyClass myObj = new MyClass(5);

System.out.println(myObj.x);

}

}

// Outputs 5

Anda dapat memiliki banyak parameter yang Anda inginkan:

Contoh

public class Car {

int modelYear;

String modelName;

public Car(int year, String name) {

modelYear = year;

modelName = name;

}

public static void main(String[] args) {

Car myCar = new Car(1969, "Mustang");

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

}

}

// Outputs 1969 Mustang

* 1. **Java Modifier**

Sekarang, Anda cukup akrab dengan kata kunci **public**yang muncul di hampir semua contoh kami:

**public** class MyClass

Kata kunci **public** adalah **pengubah akses** , artinya digunakan untuk mengatur tingkat akses untuk kelas, atribut, metode, dan konstruktor.

Kami membagi pengubah menjadi dua kelompok:

* **Access Modifiers** - mengontrol level akses.
* **Non-Access Modifiers** - tidak mengontrol level akses, tetapi menyediakan fungsionalitas lain
* Access Modifiers.
* Untuk **kelas** , Anda dapat menggunakan *public* atau default :
* **Modifiers** : Public

**Deskripsi** : kelas ini diakses oleh lain kelas

Contoh :

public class MyClass {

public static void main(String[] args) {

System.out.println("Hello World");

}

}

Output:

Hello World

* **Modifiers :** Default

**Deskripsi :** Kelas ini hanya diakses oleh kelas di sama paket. Ini adalah digunakan ketika anda tidak specify sebuah modifier. Anda akan belajar lebih banyak tentang packages di packages bab.

Contoh :

class MyClass {

public static void main(String[] args) {

System.out.println("Hello World");

}

}

Output:

Hello World

* Untuk **atribut, metode, dan konstruktor** , Anda dapat menggunakan salah satu dari yang berikut ini:
* **Modifiers** : Public

**Deskripsi**  : Kode adalah di akses untuk semua kelas

Contoh :

class MyClass {

public static void main(String[] args) {

Person myObj = new Person();

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

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

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

}

}

Output :

Name: John Doe  
Email: john@doe.com  
Age: 24

* **Modifiers** : Private

**Deskripsi**  : Kode ini m,hanya diakses dalam menyatakan kelas

Contoh :

public class Person {

private String fname = "John";

private String lname = "Doe";

private String email = "john@doe.com";

private int age = 24;

public static void main(String[] args) {

Person myObj = new Person();

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

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

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

}

}

Output:

Name: John Doe  
Email: john@doe.com  
Age: 24

* **Modifiers** : Default

**Deskripsi**  : Kode tidak hanya diakses disama paket. Ini adalah digunakan ketika anda tidah spesifysebuah modifier. Anda akan belajar lebih banyak tentang packages di sebuah packages bab

Contoh :

class Person {

String fname = "John";

String lname = "Doe";

String email = "john@doe.com";

int age = 24;

public static void main(String[] args) {

Person myObj = new Person();

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

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

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

}

}

Output:

Name: John Doe  
Email: john@doe.com  
Age: 24

* **Modifiers** : Protected

**Deskripsi**  : Kode tidak hanya diakses disama paket dan subclasses. Anda akan belajar lebih banyak tentang sublasses dan sup~erclasses di sebuah inheritance bab.

Contoh :

class Person {

protected String fname = "John";

protected String lname = "Doe";

protected String email = "john@doe.com";

protected int age = 24;

}

class Student extends Person {

private int graduationYear = 2018;

public static void main(String[] args) {

Student myObj = new Student();

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

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

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

System.out.println("Graduation Year: " + myObj.graduationYear);

}

}

## Output:

Name: John Doe  
Email: john@doe.com  
Age: 24  
Graduation Year: 2018

## Non-Access Modifiers

* Untuk **kelas** , Anda dapat menggunakan final atau abstract :

|  |  |
| --- | --- |
| **Modifiers** | **Deskripsi** |
| Final | Kelas tidak dapat diwarisi oleh kelas lain (Anda akan belajar lebih banyak tentang warisan dalam bab Warisan) |
| Abstract | Kelas tidak dapat digunakan untuk membuat objek (Untuk mengakses kelas abstrak, itu harus diwarisi dari kelas lain. Anda akan belajar lebih banyak tentang pewarisan dan abstraksi di bab Warisan dan Abstraksi) |

* Untuk **atribut dan metode** , Anda dapat menggunakan salah satu dari yang berikut:

|  |  |
| --- | --- |
| **Modifiers** | **Deskripsi** |
| Final | Atribut dan metode tidak dapat diganti atau dimodifikasi |
| Static | Atribut dan metode milik kelas, bukan objek |
| Abstract | Hanya dapat digunakan dalam kelas abstrak, dan hanya dapat digunakan pada metode. Metode ini tidak memiliki tubuh, misalnya abstrak void run (); . Tubuh disediakan oleh subclass (diwarisi dari). Anda akan belajar lebih banyak tentang warisan dan abstraksi dalam bab-bab Warisan dan Abstraksi |
| Transient | Atribut dan metode dilewati saat membuat serialisasi objek yang memuatnya |
| Synchronized | Metode hanya dapat diakses oleh satu utas pada satu waktu |
| Volatile | Nilai atribut tidak di-cache thread-lokal, dan selalu dibaca dari "memori utama" |

## Keyword Final

Jika Anda tidak ingin kemampuan untuk menimpa nilai atribut yang ada, nyatakan atribut sebagai final :

**Contoh :**

public class MyClass {

**final** int x = 10;

**final** double PI = 3.14;

public static void main(String[] args) {

MyClass myObj = new MyClass();

// myObj.x = 50; // will generate an error: cannot assign a value to a **final** variable

// myObj.PI = 25; // will generate an error: cannot assign a value to a **final** variable

System.out.println(myObj.x);

System.out.println(myObj.PI);

}

}

Output:  
10

3,14

* Statis

Metode static berarti dapat diakses tanpa membuat objek kelas, tidak seperti public :

Contoh

Contoh untuk menunjukkan perbedaan antara metode static dan public :

public class MyClass {

// Static method

static void myStaticMethod() {

System.out.println("Static methods can be called without creating objects");

}

// Public method

public void myPublicMethod() {

System.out.println("Public methods must be called by creating objects");

}

// Main method

public static void main(String[ ] args) {

myStaticMethod(); // Call the static method

// myPublicMethod(); This would output an error

MyClass myObj = new MyClass(); // Create an object of MyClass

myObj.myPublicMethod(); // Call the public method

}

}

Output:

Static methods can be called without creating objects  
Public methods must be called by creating objects

## Abstrak

Metode abstract milik kelas abstract , dan tidak memiliki tubuh. Tubuh disediakan oleh subclass:

Contoh:

// Code from filename: Person.java

// abstract class  
abstract class Person {

public String fname = "John";

public int age = 24;

public **abstract** void study(); // abstract method

}

// Subclass (inherit from Person)

class Student extends Person {

public int graduationYear = 2018;

public void study() { // the body of the abstract method is provided here

System.out.println("Studying all day long");

}

}

// End code from filename: Person.java

// Code from filename: MyClass.java

class MyClass {

public static void main(String[] args) {

// create an object of the Student class (which inherits attributes and methods from Person)

Student myObj = new Student();

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

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

System.out.println("Graduation Year: " + myObj.graduationYear);

myObj.study(); // call abstract method  
 }

}

Output:

Name: John  
Age: 24  
Graduation Year: 2018  
Studying all day long

* 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 variable
* 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 nilai.

Sintaks untuk keduanya adalah bawa dimulai dengan salah satu get atau set , diikuti oleh nama variabel , dengan huruf pertama menggunakan huruf kapital.

Berikut adalah Contoh dari encapsulation

Disini kita membuat 2 kelas yaitu kelas ‘main’ (1) & kelas yang menampilkan hasil ‘output’ (2)

public class Person {

private String name; // private = restricted access

// Getter

public String getName() {

return name;

}

// Setter

public void setName(String newName) {

this.name = newName;

}

}

Metode Set artinya mengambil parameter (newName) dan mengisi data ke dalam objek ,kata this merupakan kunci yang digunakan untuk merujuk ke objek saat ini.

Variabel ‘nama’ dideklarasikan sebagai ‘public’ agar dapat mengaksesnya di luar kelas sedangkan jika kita mendeklarasikan dengan ‘private’ maka tidak dapat diakses diluar kelas.

Tujuan dari Encapsulation antara lain :

* mengamankan atribut dan metode kelas dengan baik
* Atribut kelas dapat dibuat hanya-baca (jika hanya menggunakan get method) atau hanya-menulis (jika hanya menggunakan set method)
* Fleksibel artinya, Programmer dapat mengubah satu bagian kode tanpa mempengaruhi bagian yang lainnya.
* Peningkatan terhadap keamanan data.
  1. **Java API/ Packages**

Paket di Java digunakan untuk mengelompokkan kelas terkait. Anggap saja sebagai **folder dalam direktori file** . Kami menggunakan paket untuk menghindari konflik nama, dan untuk menulis kode yang dapat dikelola dengan lebih baik. Paket dibagi menjadi dua kategori:

* Paket Built-in (paket dari Java API)
* Paket Buatan Pengguna (buat paket Anda sendiri)
* Paket Built-in (paket dari Java API)

Java API adalah pustaka kelas yang sudah ditulis sebelumnya, yang bebas digunakan, termasuk dalam Java Development Environment.

Perpustakaan berisi komponen untuk mengelola input, pemrograman basis data, dan banyak lagi lainnya. Daftar lengkap dapat ditemukan di situs web Oracles: *[https://docs.oracle.com/javase/8/docs/api/](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://docs.oracle.com/javase/8/docs/api/&xid=17259,15700021,15700186,15700190,15700256,15700259,15700262,15700265,15700271,15700283&usg=ALkJrhiQjj-j2JCLVpyCTQLdaKKME_87ng" \t "_blank)* .

Perpustakaan dibagi menjadi beberapa **paket** dan **kelas** . Berarti Anda dapat mengimpor satu kelas (beserta metode dan atributnya), atau seluruh paket yang berisi semua kelas yang termasuk dalam paket yang ditentukan.

Untuk menggunakan kelas atau paket dari perpustakaan, Anda perlu menggunakan kata kunci **import** :

import *package*.*name*.*Class*; // Import a single class

import *package*.*name*.\*; // Import the whole package

## Impor Kelas

## Class yang bisa kita digunakan untuk mengambil input dari user, yaitu *Scanner,* *BufferedReader* dan *JOptionPane*, ketiga class tersebut mempunyai fungsi yang sama, akan tetapi pada JOptionPane, class ini memiliki tampilan visual atau GUI.

* **Class Scanner**

Untuk menggunakan Scanner, kita perlu membuat objek dari class Scanner, class tersebut berada didalam package *java.util.\**, jadi kita perlu menginport packagenya terlebih dahulu, setelah package dan objek selesai dibuat, selanjutnya kita membutuhkan sebuah variabel untuk menampung nilai yang di inputkan dari user

* + **Menyimpan Data**

Untuk menyimpan data String, kita bisa menggunakan method *next()* atau *nextLine()* pada Scanner, selain tipe data String, ada beberapa method lainnya yang bisa kalian gunakan pada Scanner, diantaranya:

**nextInt** - Menerima input bilangan bulat/Integer

**nextBoolean** - Menerima input Boolean (true/false)

**nextFloat**  - Menerima input pecahan/Float

**nextDouble**  - Menerima input pecahan/Double

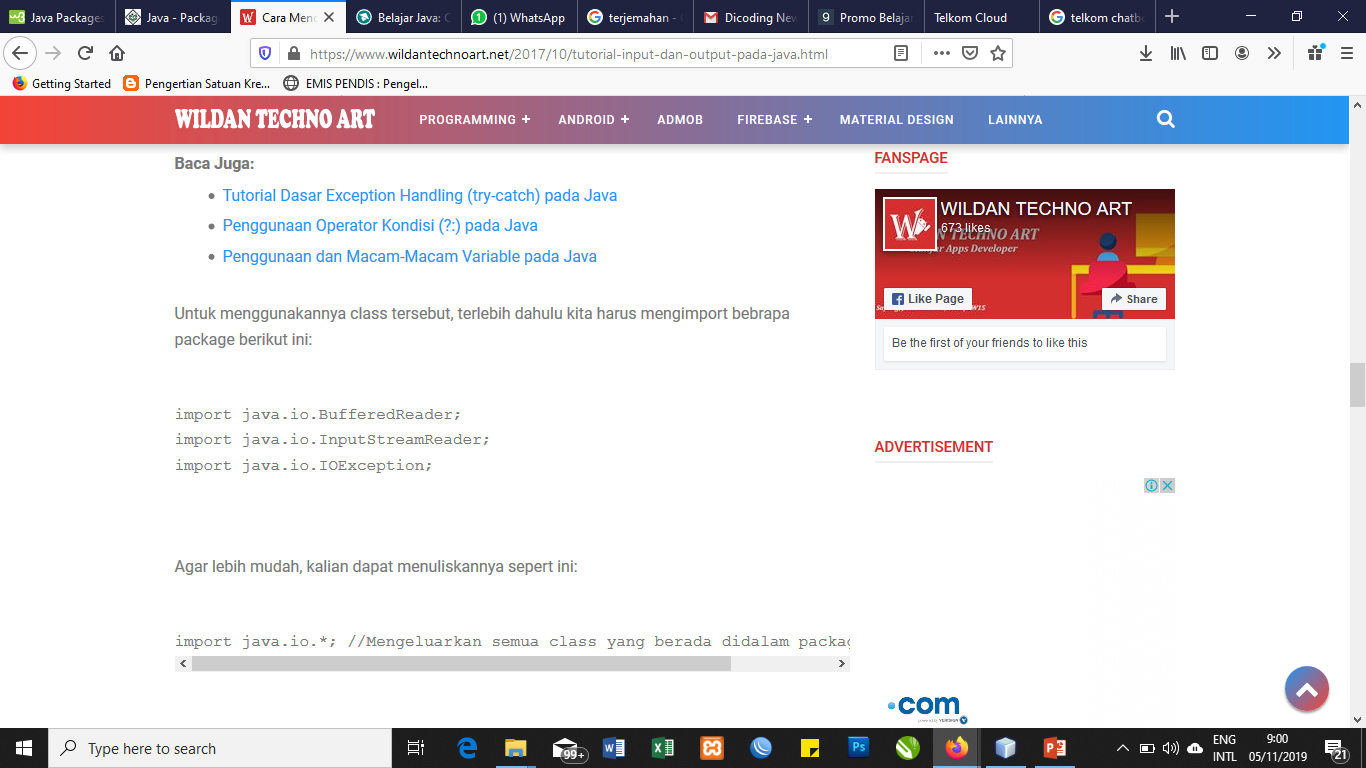
**nextByte** - Menerima bilangan bulat/Byte

**nextLong**  - Menerima bilangan bulat/Long

**nextShort**  - Menerima bilangan bulat/Short

* **Buffer Reader**

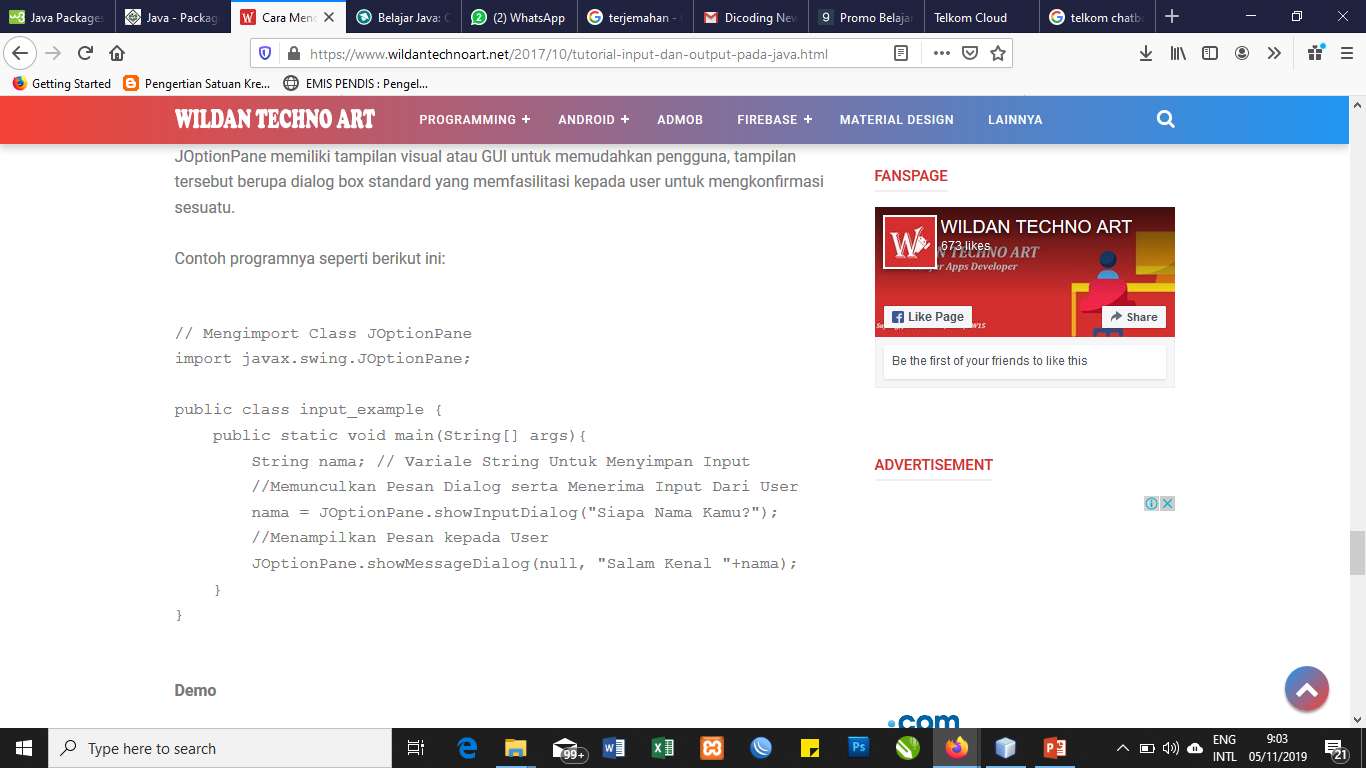
Pada bagian ini, kita akan menggunakan class BufferedReader, class tersebut berada di dalam package *java.io*, class ini dignakan untuk mendapatkan input dari user, hampir sama seperti Scanner, yang membedakan hanya cara penulisan syntaxnya saja.



* **Class JOptionPane**

Cara terakhir untuk mendapatkan input dari user adalah dengan menggunakan class JOptionPane calss tersebut terdapat pada package *javax.swing*. berbeda dari class sebelunya, JOptionPane memiliki tampilan visual atau GUI untuk memudahkan pengguna, tampilan tersebut berupa dialog box standard yang memfasilitasi kepada user untuk mengkonfirmasi sesuatu.

Contoh :



* **Class Console**

Class Console hampir sama dengan BufferReader. Dia juga menggunakan fungsi readLine() untuk mengambil input.

Akan tetapi…

Class ini hanya bisa digunakan di lingkungan console saja, seperti Terminal dan CMD.

Class Console tidak bisa digunakan langsung di Netbeans.

Maka dari itu, kita harus kompilasi secara manual.

Untuk menggunakan class ini, kita perlu mengimpornya terlebih dahulu.

* 1. **Java Inheritance**

**Inheritance** (Pewarisan) adalah salah satu contoh [Konsep OOP](http://www.okedroid.com/search/label/Konsep%20OOP?max-results=5" \t "_blank), yang merupakan suatu proses dimana , suatu class yang bisa disebut *super class* ,dapat mewarisi sifat atau ciri-ciri,  seperti atribut dan method, ke dalam class turunanya yaitu *sub class*. Super class akan mewarisi nilai dari atribut atau behavior ke Class turunanya atau class lain. Sistem penurunan sifatnya akan dilakukan secara hirarki (bertingkat-tingkat) , semakin ke bawah class, maka akan semakin detail.

* **subclass** (anak) - kelas yang mewarisi dari kelas lain.
* **superclass** (orang tua) - kelas yang diwarisi dari

Untuk mewarisi dari kelas, gunakan keyword extends yaitu sebagai kata kunci. Keyword **extends** adalah salah satu [jenis keyword](https://www.okedroid.com/2017/09/belajar-mengenal-jenis-keyword-pada-java.html) , di dalam bahasa pemrograman java, yang di gunakan untuk mewarisi sifat-sifat dari class parent atau super, seperti atribut dan method, ke dalam suatu class turunan.

Contoh :

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 :

Tuut, tuut!  
Ford Mustang

* Keyword Final

Ketika suatu class dideklarasikan sebagai final maka class tersebut tidak bisa diwariskan atau mengakibatkan output menjadi eror. Beberapa tujuan dari kelas FINAL:

Ada tiga tujuan utama kenapa suatu class dibuat final. Tujuan tersebut adalah:

1. Untuk mencegah inheritance karena class final tidak bisa di extends.
2. Untuk membuat suatu class immutable. Class tidak akan bisa dibuat immutable jika sebelumnya class itu tidak final.
3. Class yang dibuat final sangat berguna ketika akan digunakan untuk tujuan tingkat keamanan tinggi dalam setiap aplikasi.

Contoh:

final class Vehicle {

...

}

class Car extends Vehicle {

...

}

Output :

Car.java:8: error: cannot inherit from final Vehicle  
class Car extends Vehicle {  
                  ^  
1 error

* Jadi meng-exetend-kan class final akan menyebabkan error
* Ketika suatu class dideklarasikan sebagai final maka class tersebut tidak bisa diwariskan.
* Misalnya pada package diatas anda bisa melihat jika **[class String](http://bahasajava.com/belajar-java-class-string-fundamental-dan-method/)** adalah class final. Maksudnya class String tidak dapat dideklarasikan menjadi kelas final.
  1. **Java Polymorishm**

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.

Konsep dari Polymorphism menggunakan 2 metode. Metode pertama yaitu Pewarisan yang menggunakan konsep overriding yaitu membuat method yang bernama dan berstruktur sama akan tetapi isi method berbeda pada class induk dan class anak. Metode yang kedua adalah overloading. Overloading adalah method bernama yang sama dengan method lain pada suatu class tetapi dengan parameter da nisi yang berbeda.

Contoh :

class Animal {

public void animalSound() {

System.out.println("The animal makes a sound");

}

}

class Pig extends Animal {

public void animalSound() {

System.out.println("The pig says: wee wee");

}

}

class Dog extends Animal {

public void animalSound() {

System.out.println("The dog says: bow wow");

}

}

class MyMainClass {

public static void main(String[] args) {

Animal myAnimal = new Animal(); // Create a Animal object

Animal myPig = new Pig(); // Create a Pig object

Animal myDog = new Dog(); // Create a Dog object

myAnimal.animalSound();

myPig.animalSound();

myDog.animalSound();

}

}

Output :

The animal makes a sound  
The pig says: wee wee  
The dog says: bow wow

* Keuntungan Polymorphism:
* Kita dapat menggunakan kelas-kelas yang kita buat (sebagai super kelas) dan membuat kelas kelas baru berdasar superkelas tersebut dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki superkelas.
* kita dapat membuat super kelas yang hanya mendefinisikan behaviuor namun tidak memberikan implementasidari metode-metode yang ada.Hal ini berguna jika kita ingin membuat semacam template kelas, kelas semacam ini disebut kelas abstrak karena behaviuor nya masih abstrak dan belum di implementasikan. Subkelas - subkelas dari kelas semacam ini yang disebut kelas konkret,mengimplementasikan behaviuor abstrak tersebut sesuai dengan kebutuhan masing-masing.
* Menghindari duplikasi object,kita dapat menciptakan class baru dari class yang sudah ada,sehingga tidak perlu menuliskan code dari nol ataupun mengulangnya, namun tetap bisa menambahkan attribute dan atau method unik dari class itu sendiri. Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu interface banyak aksi.

**2.11 Java Inner Classes**

Di dalam java, inner class merupakan kelas besarang (kelas dalam kelas). Tujuan dari inner class untuk mengelompokkan kelas-kelas yang termasuk bersama, yang membuat sebuah source code lebih mudah dibaca dan di pelihara.

Untuk mengakses inner class, buatlah object dari kelas luar, dan kemudian buat objek dari kelas dalam. Berikut adalah contohnya:

class OuterClass {

int x = 10;

class InnerClass {

int y = 5;

}

}

public class MyMainClass {

public static void main(String[] args) {

OuterClass myOuter = new OuterClass();

OuterClass.InnerClass myInner = myOuter.new InnerClass();

System.out.println(myInner.y + myOuter.x);

}

}

Output:

15

* Private Inner Class

Tidak seperti kelas regular, kelas dalam bisa **private**  atau **protected.** Jika tidak ingin object luar mengakses kelas dalam, maka nyatakan dalam kelas sebagai **private**. Seperti contoh dibawah ini:

class OuterClass {

int x = 10;

**private** class InnerClass {

int y = 5;

}

}

public class MyMainClass {

public static void main(String[] args) {

OuterClass myOuter = new OuterClass();

OuterClass.InnerClass myInner = myOuter.new InnerClass();

System.out.println(myInner.y + myOuter.x);

}

}

Jika dilihat dalam gambar, maka muncul garis merah pada syntax yang salah. Dan jika tetap ingin mengakses private inner class dari kelas luar (MyMainClass), maka terdapat kesalahan (error):

MyMainClass.java:12: error: OuterClass.InnerClass has private access in OuterClass  
OuterClass.InnerClass myInner = myOuter.new InnerClass();  
              ^

* Static Inner Class

Kelas dalam juga bisa menggunakan static, yang berarti bahwa dapat mengaksesnya tanpa membuat objek dari kelas luar. Berikut adalah contoh dari source kode menggunakan static:

class OuterClass {

int x = 10;

static class InnerClass {

int y = 5;

}

}

public class MyMainClass {

public static void main(String[] args) {

OuterClass.InnerClass myInner = new OuterClass.InnerClass();

System.out.println(myInner.y);

}

}

Output :

5

Perlu diingat bahwa, sama seperti **static** atribut dan metode, **static**  kelas dalam tidak memiliki akses ke anggota kelas kelas luar.

* Access Outer Class dari Inner Class

Terdapat satu keuntungan dari adanya kelas dalam, adalah mereka dapat mengakses atribut dan metode dari kelas luar, contoh gambar dibawah ini merupakan source code yang dapat mengakses kelas luar dari kelas dalam:

class OuterClass {

int x = 10;

class InnerClass {

public int myInnerMethod() {

return x;

}

}

}

public class MyMainClass {

public static void main(String args[]) {

OuterClass myOuter = new OuterClass();

OuterClass.InnerClass myInner = myOuter.new InnerClass();

System.out.println(myInner.myInnerMethod());

}

}

Output :

10

**2. 12 Java Abstraction**

Abstraksi data adalah proses menyembunyikan detail tertentu dan hanya menampilkan informasi penting kepada pengguna. Abstraksi dapat dicapai dengan kelas atau interface abstrak.

Kata kunci abstract adalah pengubah non-akses, digunakan untuk kelas dan metode:

1. Kelas abstrak : adalah kelas terbatas yang tidak dapat digunakan untuk membuat objek (untuk mengaksesnya, itu harus diwarisi dari kelas lain).
2. Metode abstrak : hanya dapat digunakan dalam kelas abstrak, dan tidak memiliki tubuh, tubuh disediakan dari subclass(pewarisan dari).

Kelas abstrak dapat memiliki metode abstrak dan regular, seperti gambar dibawah ini :

abstract class Animal {

public abstract void animalSound();

public void sleep() {

System.out.println("Zzz");

}

}

Dari contoh diatas, tidak mungkin membuat objek dari kelas Animal:

Animal myObj = new Animal(); // will generate an error

Untuk mengakses kelas abstrak, harus diwarisi dari kelas lain. Maka dari itu, konversikan kelas Animal pada bab polymorphism menjadi kelas abstrak. Kata kunci yang digunakan pada bab inheritance saat mewarisi dari kelas adalah **extends.**

// Abstract class

abstract class Animal {

// Abstract method (does not have a body)

public abstract void animalSound();

// Regular method

public void sleep() {

System.out.println("Zzz");

}

}

// Subclass (inherit from Animal)

class Pig extends Animal {

public void animalSound() {

// The body of animalSound() is provided here

System.out.println("The pig says: wee wee");

}

}

class MyMainClass {

public static void main(String[] args) {

Pig myPig = new Pig(); // Create a Pig object

myPig.animalSound();

myPig.sleep();

}

}

Output:

The pig says: wee wee

Zzz

* 1. **Java Interface**

Interface adalah " **kelas abstrak** " yang sepenuhnya digunakan untuk mengelompokkan metode terkait dengan benda kosong:

// interface

interface Animal {

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

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

}

Untuk mengakses metode interface , interface harus "diimplementasikan" (agak seperti diwariskan) oleh kelas lain dengan kata kunci implements (alih-alih extends ). Isi metode interface disediakan oleh kelas "implement":

// Interface

interface Animal {

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

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

}

// Pig "implements" the Animal interface

class Pig implements Animal {

public void animalSound() {

// The body of animalSound() is provided here

System.out.println("The pig says: wee wee");

}

public void sleep() {

// The body of sleep() is provided here

System.out.println("Zzz");

}

}

class MyMainClass {

public static void main(String[] args) {

Pig myPig = new Pig(); // Create a Pig object

myPig.animalSound();

myPig.sleep();

}

}

*Output:*

The pig says: wee wee  
Zzz

Catatan tentang inteface:

* Seperti **kelas abstrak** , interface **tidak** dapat digunakan untuk membuat objek (dalam contoh di atas, tidak mungkin membuat objek "Hewan" di MyMainClass)
* Metode inteface tidak memiliki tubuh - tubuh disediakan oleh kelas "implement"
* Pada implementasi interface, Anda harus mengganti semua metodenya
* Metode interface secara default abstract dan public
* Atribut interface secara default public , static dan final
* Interface tidak dapat berisi konstruktor (karena tidak dapat digunakan untuk membuat objek)

#### **Mengapa Dan Kapan Menggunakan interface ?**

1) Untuk mencapai keamanan - sembunyikan detail tertentu dan hanya tampilkan detail penting dari suatu objek (interface).

2) Java tidak mendukung "multiple inheritance" (sebuah kelas hanya dapat diwarisi dari satu superclass). Namun, ini dapat dicapai dengan interface, karena kelas dapat **mengimplementasikan** banyak interface.

* Multiple Interface

**Catatan:** Untuk mengimplementasikan beberapa interface, pisahkan dengan koma.**Contoh :**

interface FirstInterface {

public void myMethod(); // interface method

}

interface SecondInterface {

public void myOtherMethod(); // interface method

}

class DemoClass implements FirstInterface, SecondInterface {

public void myMethod() {

System.out.println("Some text..");

}

public void myOtherMethod() {

System.out.println("Some other text...");

}

}

class MyMainClass {

public static void main(String[] args) {

DemoClass myObj = new DemoClass();

myObj.myMethod();

myObj.myOtherMethod();

}

}

Output :

Some text...  
Some other text

* 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.

jika sebelumnya mendefinisikan nilai pada variable bertipe data konstanta dengan baris per baris dengan menggunakan keyword **enum**, kita dapat mengklasifikan variable-variabel bertipe data konstanta tersebut didalam **block{}**.

Untuk membuat enum, menggunakan enum kata kunci (bukan kelas atau interface), dan memisahkan konstanta dengan koma. Catatan bahwa mereka harus dalam huruf besar:

enum Level {

LOW,

MEDIUM,

HIGH

}

Anda dapat mengakses ***enum*** konstanta dengan **dot** sintaks :

Level myVar = Level.MEDIUM;

* Enum dalam pernyataan SWITCH-CASE

Merupakan pernyataan yang dirancang khusus untuk menangani suatu keputusan atau suatu hasil akhir dalam membuat project, switch-case memiliki kegunaan yang sama seperti if-else. Tetapi switch-case memiliki suatu batasan yaitu

1. Data yang diperiksa harus bertipe integer(INT) atau karakter (CHR).
2. Range data yang diperiksa bernilai 0 s/d 255.

Bentuk Penulisannya :

|  |
| --- |
| switch (value)  {  case constanta 1 : statement 1;  break;  case constanta 2 : statement 2;  break;  ……  ….etc  default : statement x;  } |

Keterangan:

* **Constanta 1, constanta 2,** dst adalah yang akan dicocokkan dengan isi value.
* **Statement** adalah pernyataan yang akan di kerjakan jika **value (bertipe Integer atau Karakter)**cocok dengan salah satu data dari **constanta 1 , constanta 2 atau yang lainnya**.
* **Break** adalah perintah untuk mengakhiri **statement**. Apabila tidak ada break, komputer akan mengeksekusi intruksi yang berada di bawahnya walaupun berada di case yang berbeda.
* **Default**, bersifat optional. dieksekusi jika **value** tidak cocok dengan salah satu **constanta**yang tersedia.
* **Loop melalui enum**

Yaitu untuk mengulang kelas enum di atas atau mengembalikan sebuah array dari semua konstanta enum. Metode values() ini berguna ketika Anda ingin loop melalui konstanta enum.

* Ada dua method yang sering digunakan pada enum atara lain
  + 1. Values ();

Method ini mengembalikan keseluruhan dari enumeration

* + 1. ValueOf(String);

Method ini mengembalikan isi dari enumeration yang sesuai dengan parameter string

bentuk penulisannya:

|  |
| --- |
| for (EnumClass obj : EnumClass.values()){  System.out.println (obj); |

* 1. **Java User Input :**

##### Input Pengguna Java

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** . Dalam contoh kita, kita akan menggunakan metode **nextLine()** , yang digunakan untuk membaca String

## Jenis Input

Pada contoh di atas, kami menggunakan metode nextLine() , yang digunakan untuk membaca Strings. Untuk membaca jenis lain, lihat tabel di bawah ini:

|  |  |
| --- | --- |
| **Method** | **Description** |
| nextBoolean() | Reads a boolean value from the user |
| nextByte() | Reads a byte value from the user |
| nextDouble() | Reads a double value from the user |
| nextFloat() | Reads a float value from the user |
| nextInt() | Reads a int value from the user |
| nextLine() | Reads a String value from the user |
| nextLong() | Reads a long value from the user |
| nextShort() | Reads a short value from the user |

* 1. **Java Date :**

***Java Date And Time***

* **Tanggal Java**

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 |

* **Menampilkan Tanggal saat ini**

Untuk menampilkan tanggal saat ini, impor kelas ***java.time.LocalDate*** dan gunakan metode ***now().***

Contoh:

|  |
| --- |
| *package date;*  *import java.time.LocalDate;* |

Pada bagian ini, kita melakukan impor Date (tanggal) yang dibutuhkan pada program.

Setelah kita mengimpor, lalu membuat kelas tanggal

|  |
| --- |
| *public class MyClass {*  *public static void main(String[] args) {*  *LocalDate myObj = LocalDate.now();* |

LocalDate myObj digunakan Untuk menambahkan objek tanggal dengan menggunakan metode Now yang artinya tanggal saat ini .

*System.out.println(myObj);*

}

}

Selanjutnya , kita memanggil objek atau menampilkan hasil outputannya .

* **Menampilkan Waktu saat ini**

Untuk menampilkan tanggal saat ini, impor kelas java.time.LocalTime dan gunakan metode now().

Contoh:

*import java.time.LocalTime;*

Pada bagian ini, kita melakukan impor Time (waktu) yang dibutuhkan pada program. Setelah itu, kita membuat kelas waktu

|  |
| --- |
| *public class MyClass {*  *public static void main(String[] args) {*  *LocalTime myObj = LocalTime.now();* |

LocalTime myObj digunakan Untuk menambahkan objek waktu dengan menggunakan metode Now yang artinya waktu saat ini .

*System.out.println(myObj);*

*}*

*}*

Selanjutnya , kita memanggil objek atau menampilkan hasil outputannya .

* **Menampilkan Tanggal dan Waktu saat ini**

Untuk menampilkan tanggal dan waktu saat ini , impor kelas java.time.LocalDateTime dan gunakan metode now().

Contoh :

*import java.time.LocalDateTime;*

pada bagian ini kita melakukan impor Date dan Time yang dibutuhkan pada program. Setelah itu, kita membuat kelas

|  |
| --- |
| *public class MyClass {*  *public static void main(String[] args) {*  *LocalDateTime myObj = LocalDateTime.now();* |

LocalDateTime myObj digunakan Untuk menambahkan objek tanggal dan waktu dengan menggunakan metode Now yang artinya tanggal dan waktu saat ini .

*System.out.println(myObj);*

*}*

*}*

.

**Tanggal dan Waktu**

‘T’ pada contoh diatas digunakan untuk memisahkan tanggal dan waktu . bisa juga menggunakan kelas DateTimeFormatter dengan metode ofPattern() dalam paket yang sama untuk memformat atau menguraikan objek waktu . Contoh berikut akan menghapus ‘T’ dan milidetik dari tanggal-waktu :

Contoh :

*import java.time.LocalDateTime;*

kita mengimpor tanggal dan waktu pada program

*import java.time.format.DateTimeFormatter;*

Selanjutnya, mengimpor format tanggal dan waktu yang akan kita atur

Setelah kita mengimpor lalu kita membuat kelas

|  |
| --- |
| *public class MyClass {*  *public static void main(String[] args) {*  *LocalDateTime myDateObj = LocalDateTime.now();* |

LocalDateTime myObj digunakan Untuk menambahkan objek tanggal dan waktu dengan menggunakan metode Now yang artinya tanggal dan waktu saat ini .

*System.out.println("Before formatting: " + myDateObj);*

Ini adalah fungsi untuk menampilkan teks ke layar output-annya dengan menambahkan objek nya.

|  |
| --- |
| *DateTimeFormatter myFormatObj =DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");* |

Selanjutnya kita mengubah format tanggal dan waktu , dengan format *ofPattern* yang menerima segala macam nilai .

*String formattedDate = myDateObj.format(myFormatObj);*

Setelah itu, menambahkan objek yang akan kita buat dalam memebrikan format yang berbeda pada tanggal dan waktu.

*System.out.println("After Formatting : " + formattedDate);*

Kita memanggil hasil outputan yang dudah di edit pada format yang di ubah.

}

}

The ofPattern() Metode menerima segala macam nilai , jika ingin menampilkan tanggal dan waktu dalam format yang berbeda. Sebagai contoh :

1.yyyy-MM-dd

*import java.time.LocalDateTime;*

*import java.time.format.DateTimeFormatter;*

*public class MyClass1 {*

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

*LocalDateTime myDateObj = LocalDateTime.now();*

*System.out.println("Before formatting: " + myDateObj);*

*DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");*

*String formattedDate = myDateObj.format(myFormatObj);*

*System.out.println("After formatting: " + formattedDate);*

*}*

*}*

Hasil Output

Before Formatting: 2019-12-22T22:01:31.570548  
After Formatting: 22-12-2019 22:01:31

2. dd/MM/yyyy

*import java.time.format.DateTimeFormatter;*

*public class MyClass2 {*

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

*LocalDateTime myDateObj = LocalDateTime.now();*

*System.out.println("Before Formatting: " + myDateObj);*

*DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");*

*String formattedDate = myDateObj.format(myFormatObj);*

*System.out.println("After Formatting: " + formattedDate);*

*}*

*}*

Hasil Output

BeforeFormatting: 2019-12-22T21:46:15.420247  
After Formatting: 22/12/2019 21:46:15

1. dd-MMM-yyyy

*import java.time.LocalDateTime;*

*import java.time.format.DateTimeFormatter;*

*public class MyClass3 {*

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

*LocalDateTime myDateObj = LocalDateTime.now();*

*System.out.println("Before Formatting: " + myDateObj);*

*DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MMM-yyyy HH:mm:ss");*

*String formattedDate = myDateObj.format(myFormatObj);*

*System.out.println("After Formatting: " + formattedDate);*

*} }*

Hasil Output

Before Formatting: 2019-12-22T21:52:55.657506  
After Formatting: 22-Dec-2019 21:52:55

1. E, MMM dd yyyy

*import java.time.LocalDateTime;*

*import java.time.format.DateTimeFormatter;*

*public class MyClass4 {*

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

*LocalDateTime myDateObj = LocalDateTime.now();*

*System.out.println("Before Formatting: " + myDateObj);*

*DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("E, MMM dd yyyy HH:mm:ss");*

*String formattedDate = myDateObj.format(myFormatObj);*

*System.out.println("After Formatting: " + formattedDate);*

*}*

*}*

Hasil Output

Before Formatting: 2019-12-22T21:58:40.868365  
After Formatting: Sun, Dec 22 2019 21:58:4

* 1. **Java Array List :**

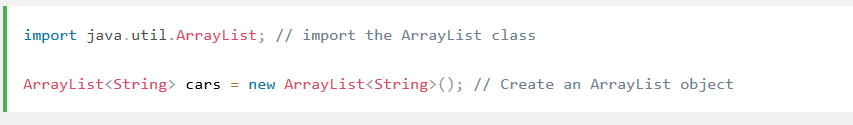
**Java Array List**

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. Sintaksnya juga sedikit berbeda:

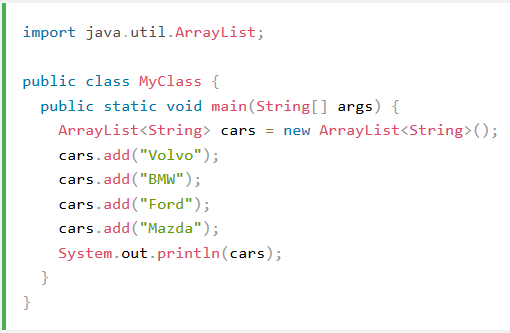
### Contoh :

Buat objek ArrayList disebut **mobil** yang akan menyimpan string:

****

* **Tambahkan Item**

Kelas ArrayList memiliki banyak metode yang berguna. Misalnya, untuk menambahkan elemen ke ArrayList , gunakan metode add() :

Contoh :

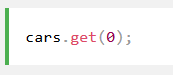
Hasil :

Run Add

1. **Akses suatu Item**

Untuk mengakses elemen di ArrayList , gunakan metode get() dan lihat nomor indeks:

Contoh :





Hasil :

Run acess

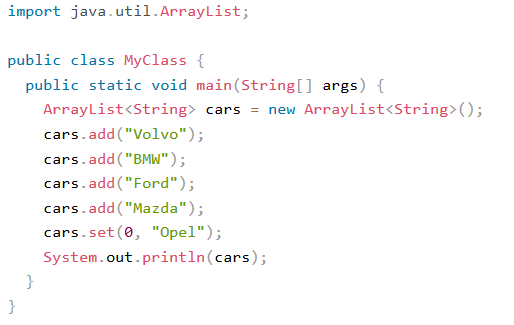
**Ingat:** Indeks array dimulai dengan 0: [0] adalah elemen pertama. [1] adalah elemen kedua, dll.

* **Ubah Item**

## Untuk memodifikasi elemen, gunakan metode set() dan lihat nomor indeks:

Contoh :





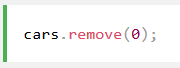
Hasil :

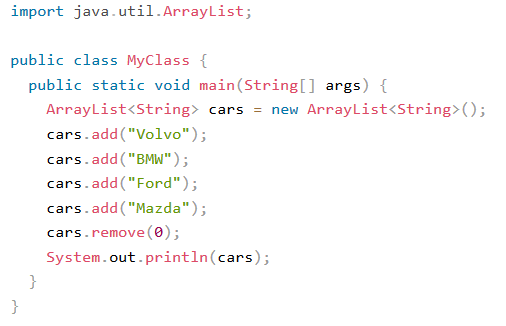
Run Change

* **Hapus Sebuah Item**

Untuk menghapus elemen, gunakan metode remove() dan lihat nomor indeks:

Contoh :

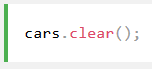


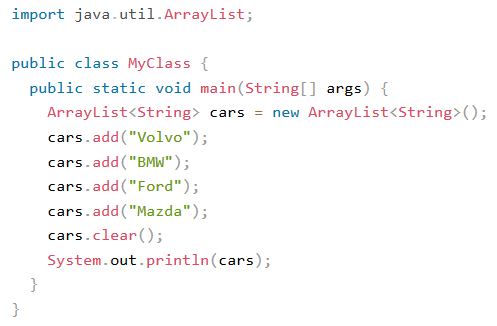


Hasil :

Run Remove

Untuk menghapus semua elemen di ArrayList , gunakan metode clear() :





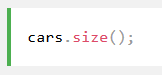
Hasil :

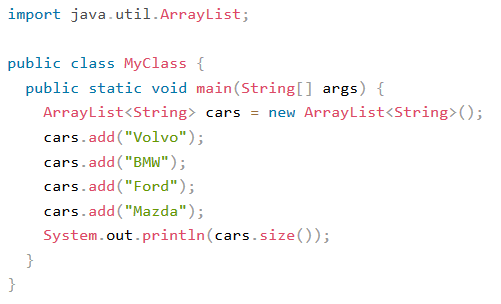
Run Clear

* **Ukuran ArrayList**

Untuk mengetahui berapa banyak elemen yang dimiliki ArrayList, gunakan metode size :

Contoh :





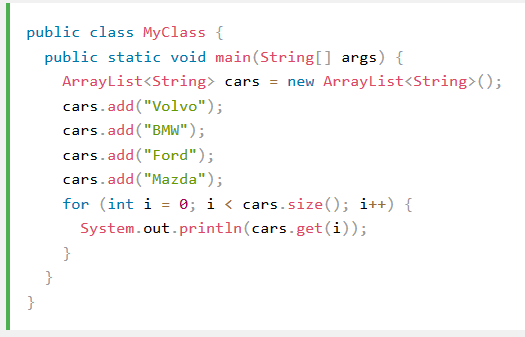
Hasil :

Run size

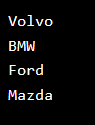
## Loop Through ArrayList

Ulangi elemen ArrayList dengan for loop, dan gunakan metode size() untuk menentukan berapa kali loop harus dijalankan:

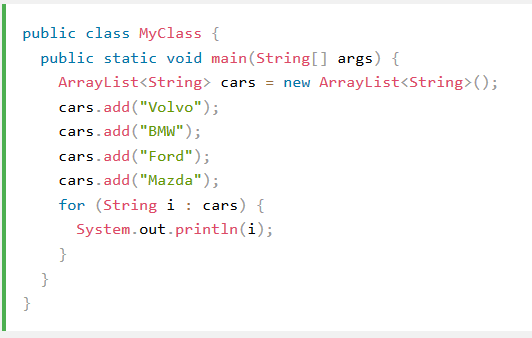
### Contoh



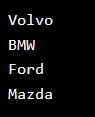
Hasil :



### Anda juga dapat mengulang melalui ArrayList dengan **untuk masing-masing** loop:



Hasil :



## Tipe yang lain

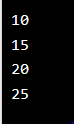
Elemen dalam ArrayList sebenarnya adalah objek. Dalam contoh di atas, kami membuat elemen (objek) dari tipe "String". Ingatlah bahwa String di Java adalah objek (bukan tipe primitif). Untuk menggunakan tipe lain, seperti int, Anda harus menentukan [kelas pembungkus yang](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.w3schools.com/java/java_wrapper_classes.asp&xid=25657,15700021,15700186,15700191,15700256,15700259,15700262,15700265,15700271,15700283&usg=ALkJrhgVuCoieuJyi6KxJTUJaORwg-c7ZQ) setara: Integer . Untuk jenis primitif lainnya, gunakan: Boolean untuk boolean, Character untuk char, Double untuk double, dll:

Contoh

Buat ArrayList untuk menyimpan angka (tambahkan elemen bertipe Integer ):



Hasil :

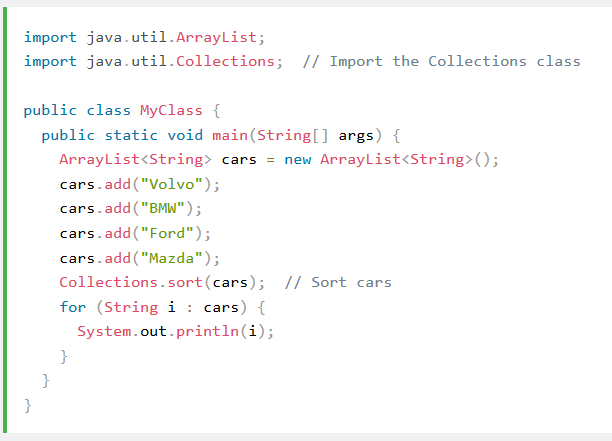


## Sortir Daftar Array

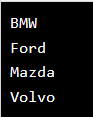
Kelas lain yang berguna dalam paket java.util adalah kelas Collections , yang mencakup metode sort() untuk menyortir daftar berdasarkan abjad atau angka:

### Contoh

Sortir ArrayList of Strings:



Hasil :

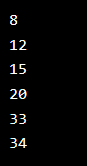


### Contoh

Sortir ArrayList of Integer:



Hasil :



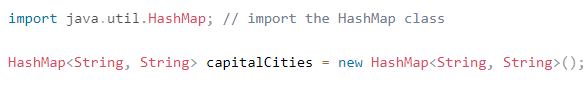
* 1. **Java HashMap :**
* **Java HashMap**

Anda belajar dari bab sebelumnya, bahwa Array menyimpan item sebagai koleksi yang dipesan, dan Anda harus mengaksesnya dengan nomor indeks (tipe int ). Namun, HashMap , menyimpan item dalam pasangan " kunci / nilai ", dan Anda dapat mengaksesnya dengan indeks jenis lain (mis. String ).

Satu objek digunakan sebagai kunci (indeks) ke objek lain (nilai). Ini dapat menyimpan berbagai jenis: Kunci String dan nilai Integer , atau tipe yang sama, seperti: Kunci String dan nilai String :

**Contoh**

Buat objek HashMap disebut capitalCities yang akan menyimpan kunci String dan nilai-nilai String :



* **Add Item**

Kelas HashMap memiliki banyak metode yang berguna. Misalnya, untuk menambahkan item ke dalamnya, gunakan metode put() :



## Access an Item

Untuk mengakses nilai di HashMap , gunakan metode get() dan lihat kuncinya:

Contoh

C:\Users\S a m s u n g\Pictures\hashmap\3.PNG

## Remove an Item

Untuk menghapus item, gunakan metode remove() dan lihat kunci:

Contoh

C:\Users\S a m s u n g\Pictures\hashmap\4.PNG

Untuk menghapus semua item, gunakan metode clear() :

Contoh

C:\Users\S a m s u n g\Pictures\hashmap\5.PNG

## Ukuran HashMap

Untuk mengetahui berapa banyak item yang ada, gunakan metode size ():

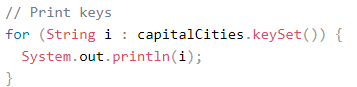
C:\Users\S a m s u n g\Pictures\hashmap\6.PNG

## Loop Through HashMap

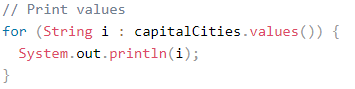
Ulangi item-item dari HashMap dengan **setiap** loop.

**Catatan:** Gunakan metode keySet() jika Anda hanya menginginkan kunci, dan gunakan metode values() jika Anda hanya menginginkan nilai:

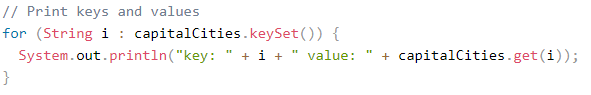
**Contoh**



Contoh



Contoh

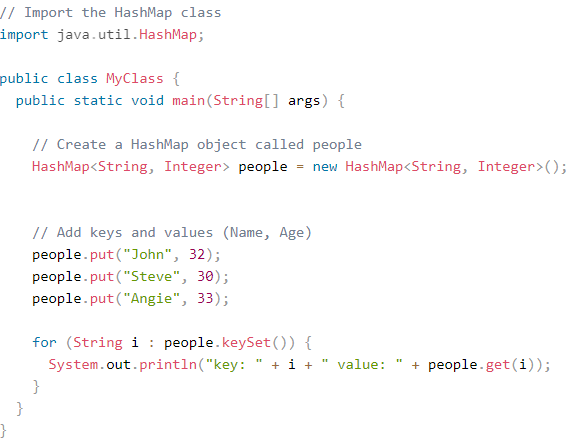


* **Tipe yang lain**

Kunci dan nilai dalam HashMap sebenarnya adalah objek. Dalam contoh di atas, kami menggunakan objek bertipe "String". Ingatlah bahwa String di Java adalah objek (bukan tipe primitif). Untuk menggunakan tipe lain, seperti int, Anda harus menentukan kelas pembungkus yang setara: Integer . Untuk jenis primitif lainnya, gunakan: Boolean untuk boolean, Character untuk char, Double untuk double, dll:

Contoh

Buat objek HashMap disebut orang yang akan menyimpan kunci String dan nilai Integer :



* 1. **Java Wrapper Classes:**

Kelas wrapper menyediakan cara untuk menggunakan tipe data primitif ( int , boolean , dll.) Sebagai objek.  
 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 |

Terkadang Anda harus menggunakan kelas wrapper, misalnya saat bekerja dengan objek Koleksi, seperti **ArrayList** , di mana tipe primitif tidak dapat digunakan (daftar hanya dapat menyimpan objek):

|  |
| --- |
| **MyClass.java**    import java.util.ArrayList;  public class MyClass {  public static void main(String[] args) {  ArrayList<Integer> myNumbers = new ArrayList<Integer>();  myNumbers.add(10);  myNumbers.add(15);  myNumbers.add(20);  myNumbers.add(25);  for (int i : myNumbers) {  System.out.println(i);  }  }  }  Output:  10 15 20 25 |

Membuat Objek Wrapper

Untuk membuat objek wrapper, gunakan kelas wrapper alih-alih tipe primitif. Untuk mendapatkan nilai, Anda cukup mencetak objek:

|  |
| --- |
| **MyClass.java**    public class MyClass {  public static void main(String[] args) {  Integer myInt = 5;  Double myDouble = 5.99;  Character myChar = 'A';  System.out.println(myInt);  System.out.println(myDouble);  System.out.println(myChar);  }  }  Output:  5  5.99  A |

Karena Anda sekarang bekerja dengan objek, Anda dapat menggunakan metode tertentu untuk mendapatkan informasi tentang objek tertentu.

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

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

Contoh ini akan menampilkan hasil yang sama seperti contoh di atas:

|  |
| --- |
| MyClass.java    public class MyClass {  public static void main(String[] args) {  Integer myInt = 5;  Double myDouble = 5.99;  Character myChar = 'A';  System.out.println(myInt.intValue());  System.out.println(myDouble.doubleValue());  System.out.println(myChar.charValue());  }  }  Output :  5 5.99 A |

Metode lain yang bermanfaat adalah metode toString() , yang digunakan untuk mengubah objek pembungkus menjadi string.

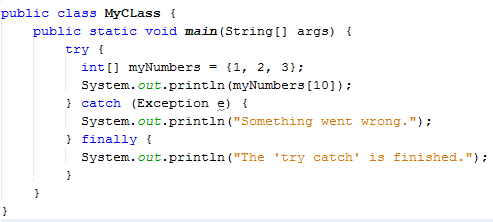
Dalam contoh berikut ini, kami mengonversi Integer ke String , dan menggunakan metode length() dari kelas String untuk menampilkan panjang "string":

|  |
| --- |
| MyClass.java  public class MyClass {  public static void main(String[] args) {  Integer myInt = 100;  String myString = myInt.toString();  System.out.println(myString.length());  }  }  Result:  3 |

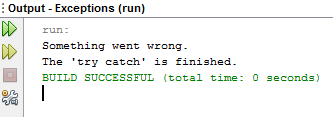
* 1. **Java Exceptions :**

Pernyataan **finally** memungkinkan Anda menjalankan kode, setelah **try...catch** , terlepas dari hasilnya:

Contoh :



Outputan:



Kata kunci lemparan

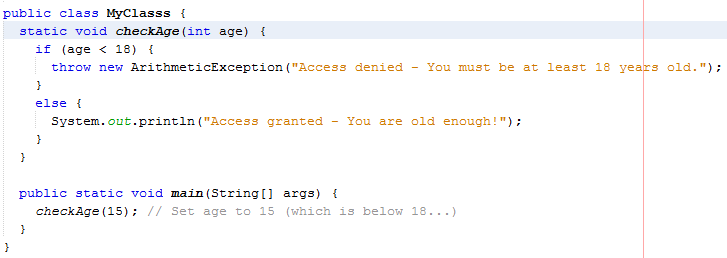
Pernyataan **throw** memungkinkan saya untuk membuat kesalahan khusus.

Pernyataan **throw** digunakan bersama dengan **tipe pengecualian** . Ada banyak jenis pengecualian yang tersedia di :

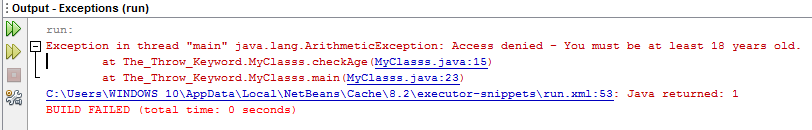
Java ArithmeticException , FileNotFoundException , ArrayIndexOutOfBoundsException , SecurityException , dll:

Contoh :

Lempar pengecualian jika usia di abwah 18 (cetak “Akses ditolak”). Jika usian 18 tahun atau lebih cetak “Akses diberika”



Outputan :

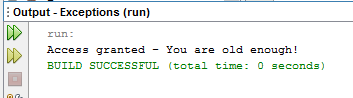


Jika usia 20 tahun, saya  tidak akan mendapatkan pengecualian:

Contoh :



Outputan:



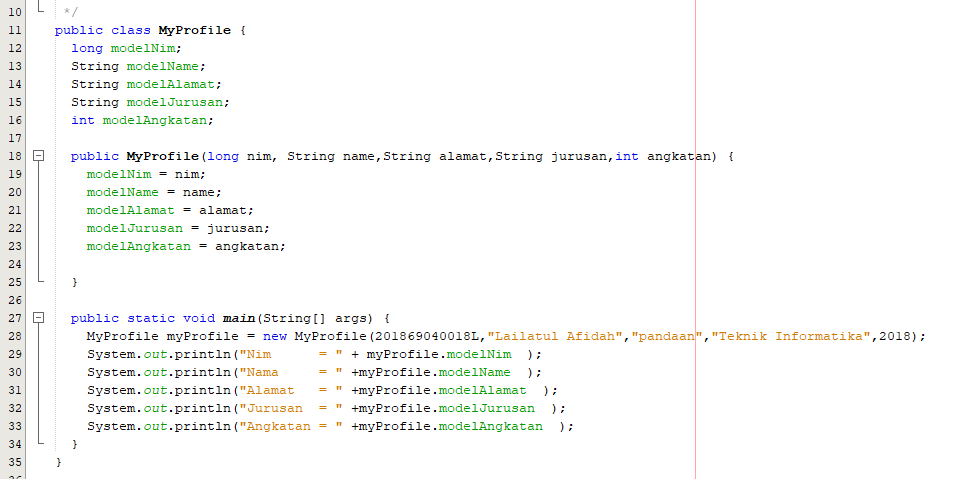
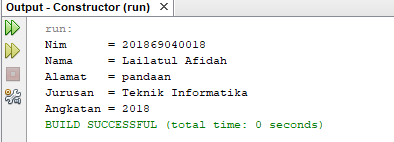
**BAB III**

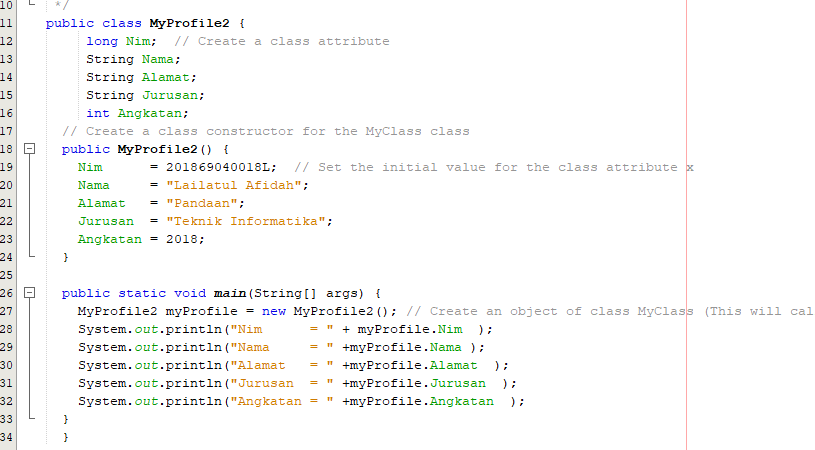
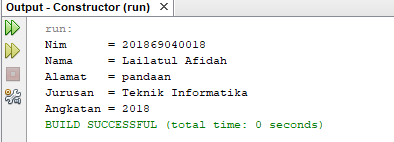
**HASIL PERCOBAAN/PRAKTIKUM**

**3.1 Hasil Percobaan 1**

Constructor

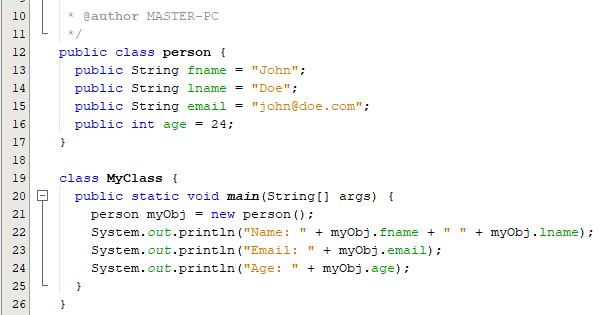
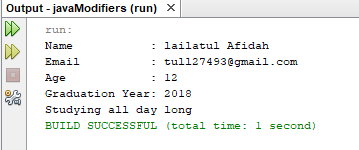
* Dengan Parameter

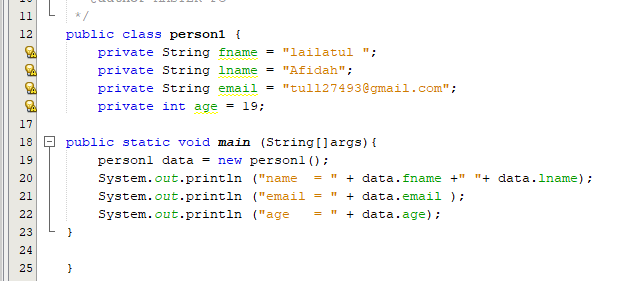


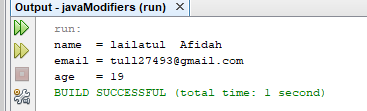
* Tanpa Parameter
  1. **Hasil Percobaan 2**

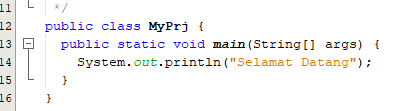
Modifiers

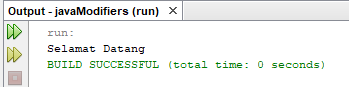
* Public



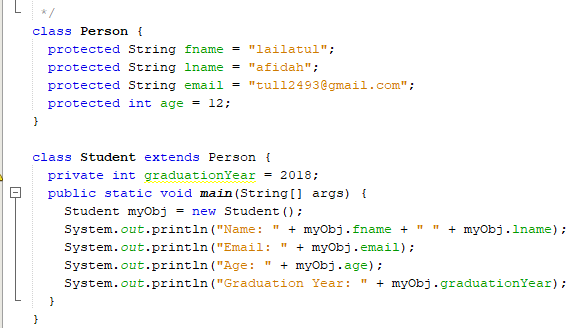
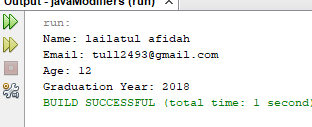
* Private



* Default

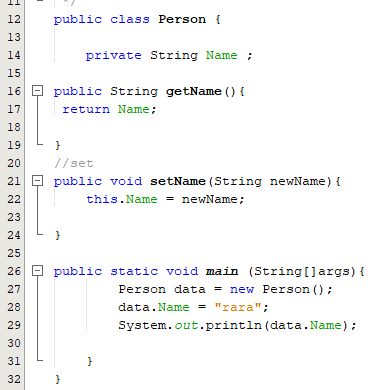


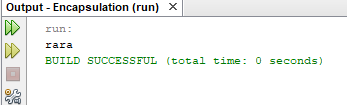
* Protected



* 1. **Hasil Percobaan 3**

Java encapsuation

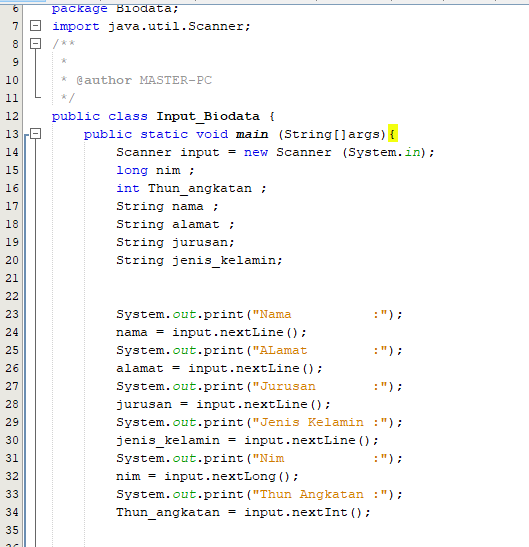
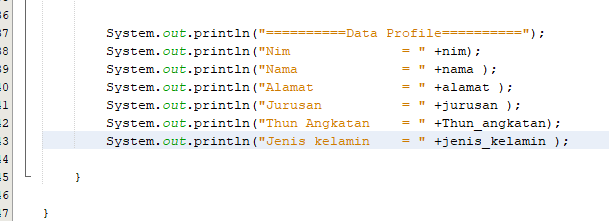




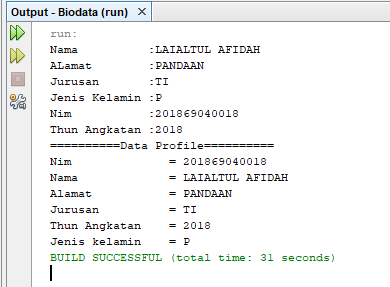
* 1. **Hasil Percobaan 4**

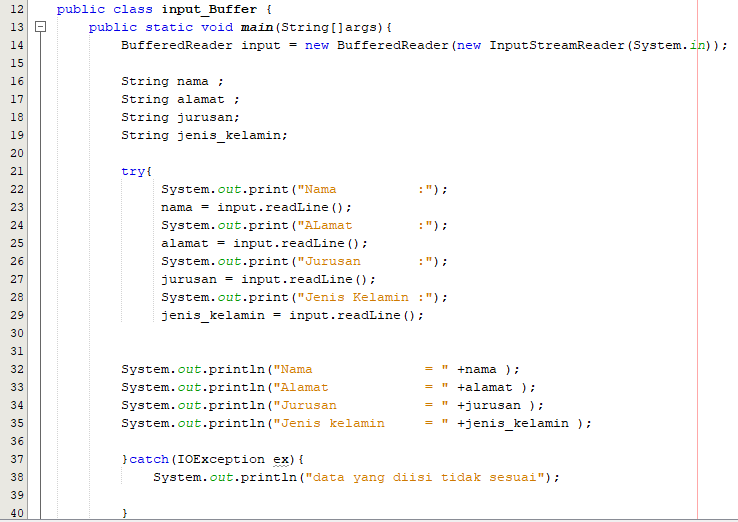
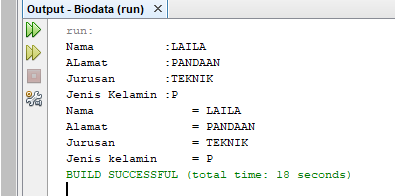
Java Package/API

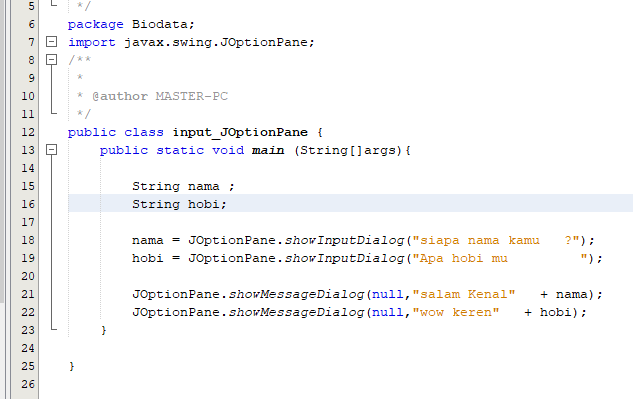
1. Scanner



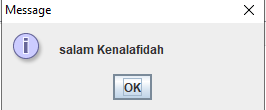
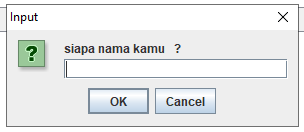
Hasil :

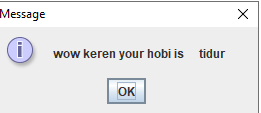
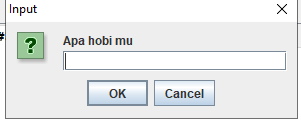


1. Buffer reader
2. jOption Pane

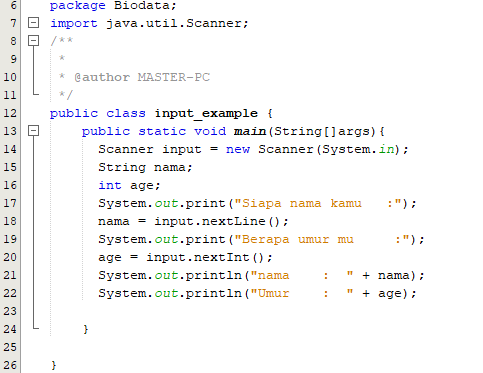


Hasil output ;

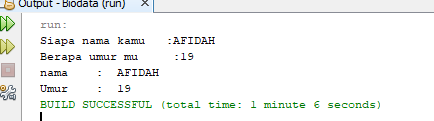
 

1. Consule

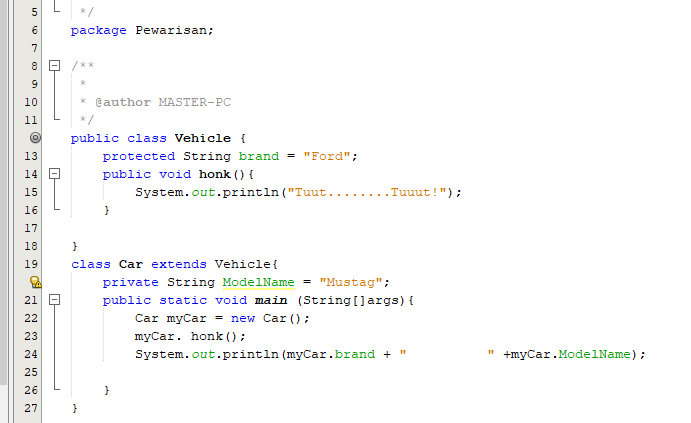


Hasil :

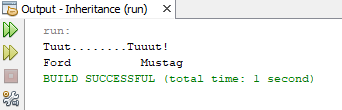


* 1. **Hasil percobaan 5**

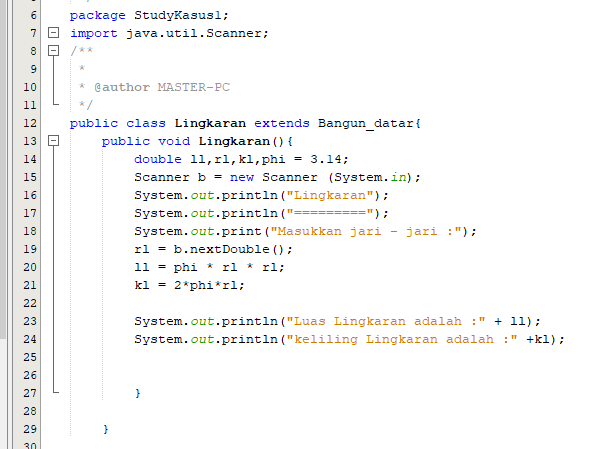
Inheritence :



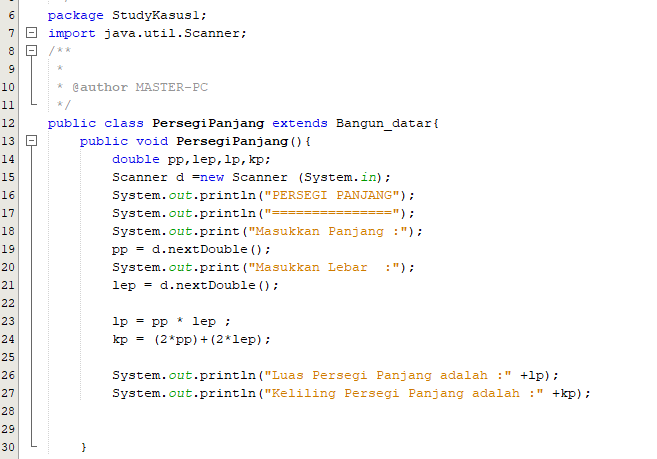
Hasil :

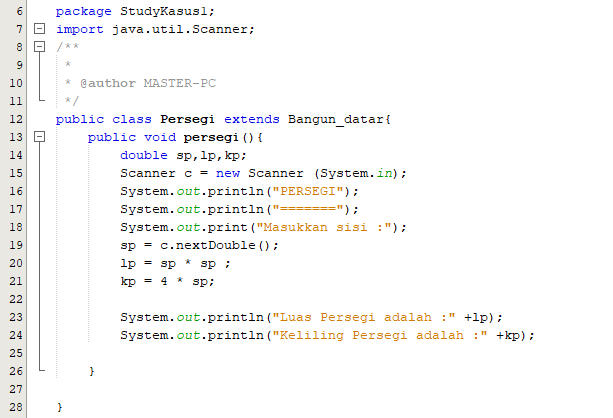


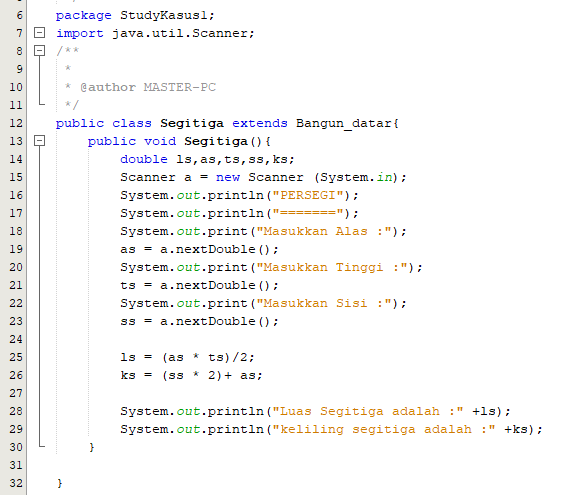
* Bangun Ruang
  + - *lingkaran*

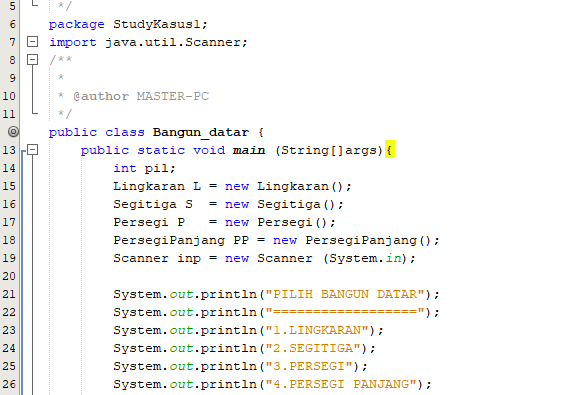


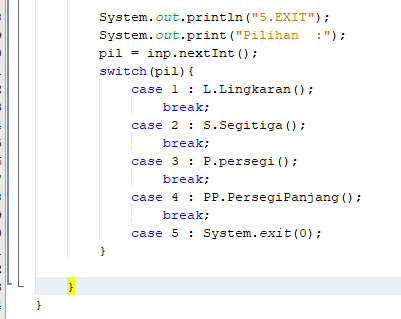
* + - *persegi panjang*



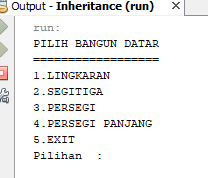
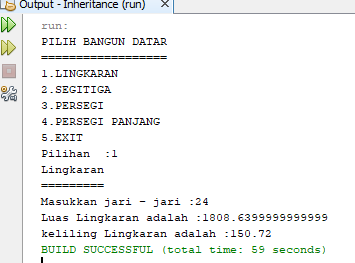


* + - *Persegi*
    - *segitiga*
    - *running file*



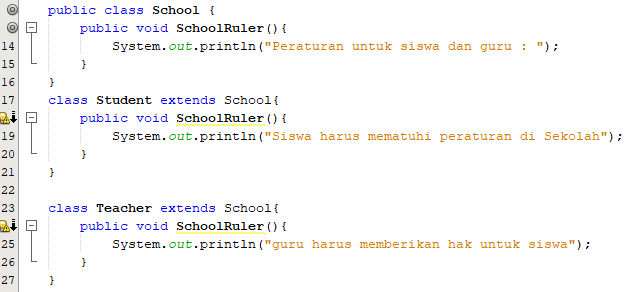


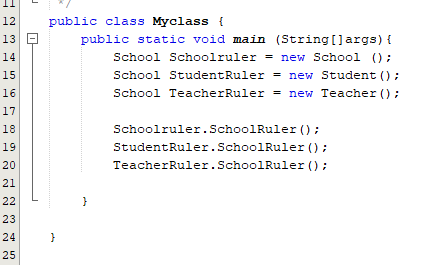
Hasil Output :



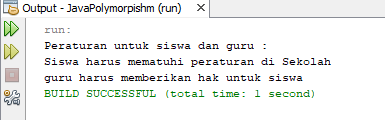
* 1. **Hasil percobaan 6**

JavaPolymorisme





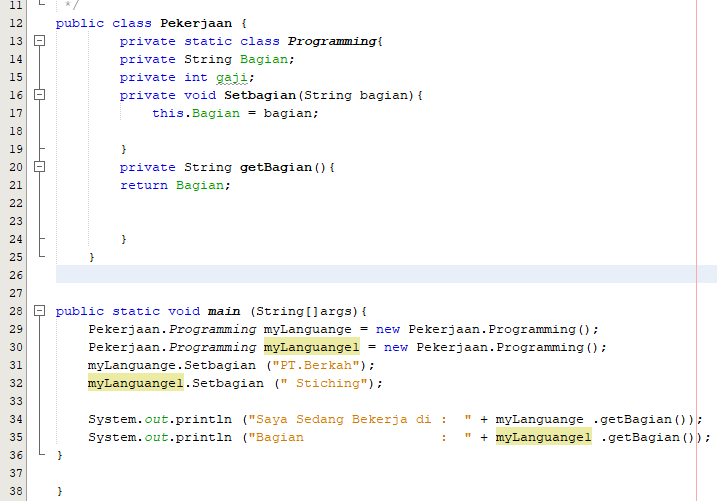
Hasil :



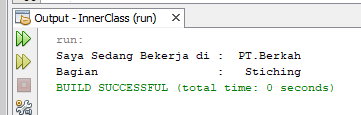
* 1. **Hasil percobaan 7**

Java Inner Class

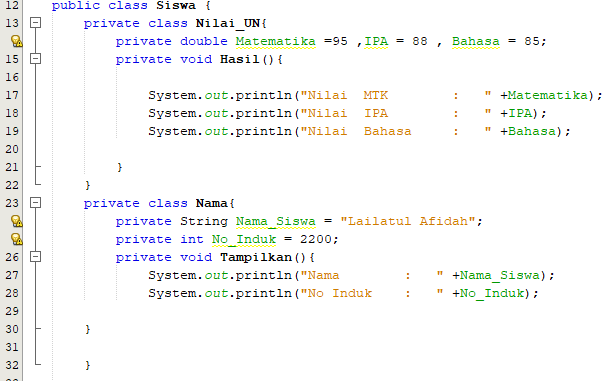
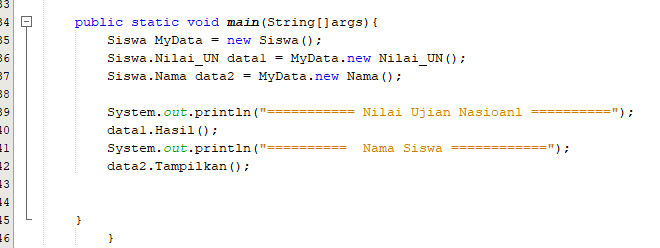
1. Static

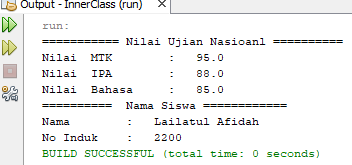


Hasil :



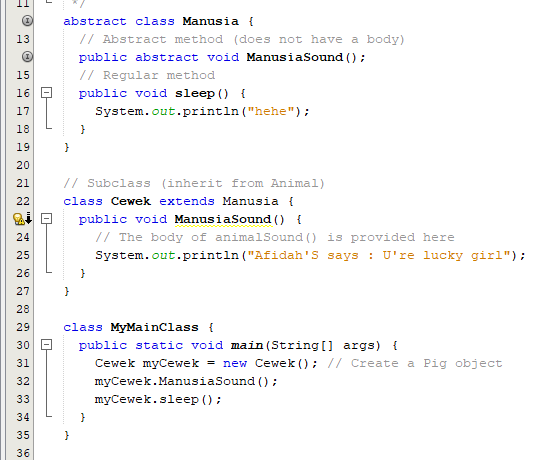
1. Non\_Static

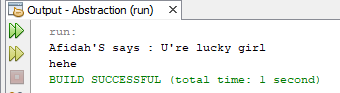


* 1. **Hasil percobaan 8**

Java Abtraction

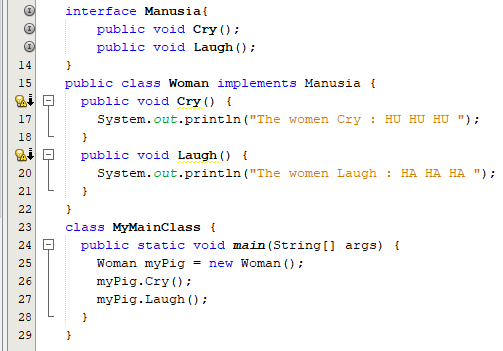


Hasil :

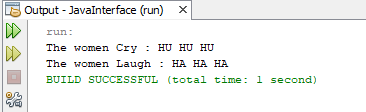


* 1. **Hasil percobaan 9**

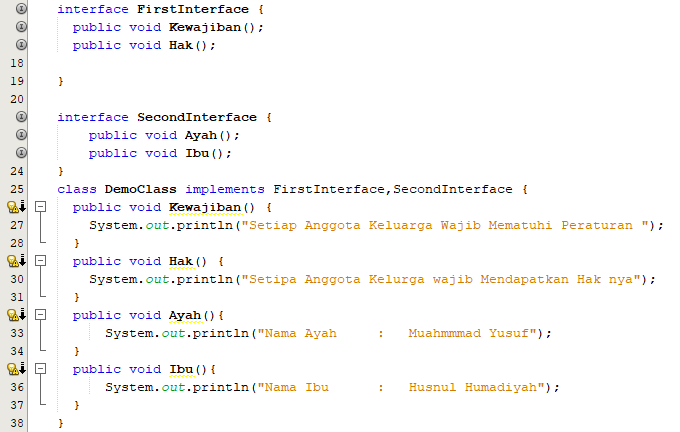
Interface

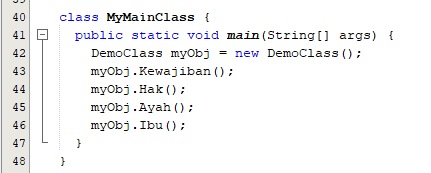


Hasil :

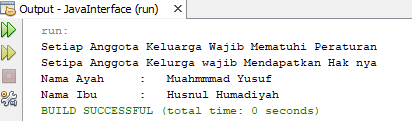


* Mulitiple



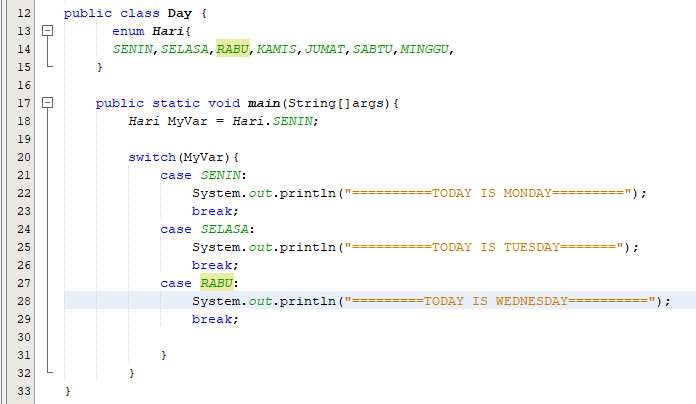


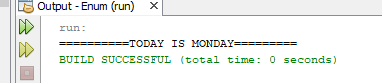
Hasil :



* 1. **Hasil Percobaan 10**

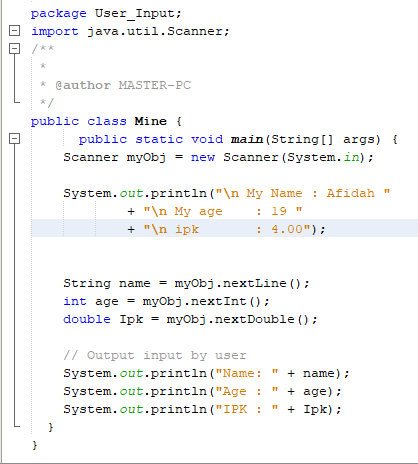
**JavaEnums**

 **Hasil :**

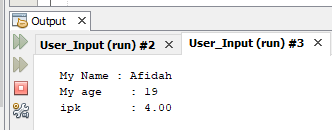


* 1. **Hasil Percobaan 11**

Java User Input

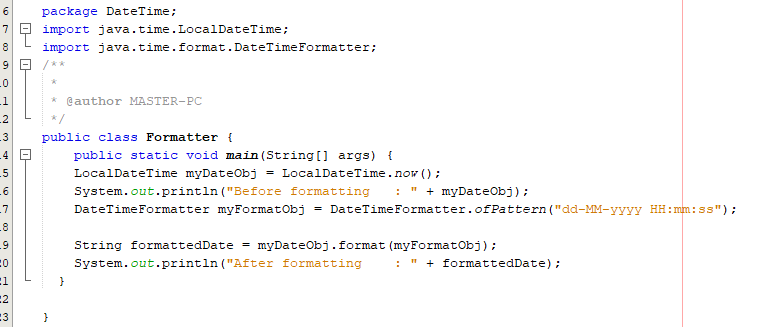


Hasil :

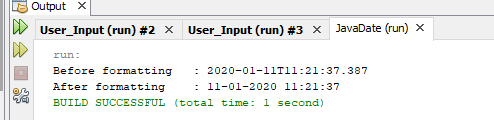


* 1. **Hasil Percobaan 12**

Java Date

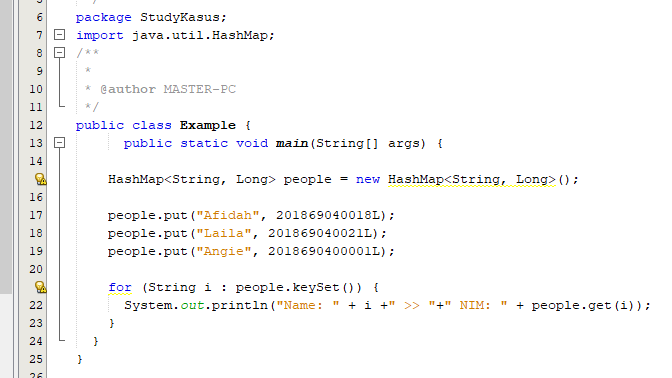


Hasil :

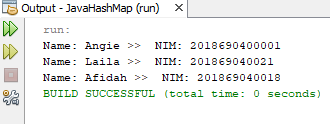


* 1. **Hasil Percobaan 13**

Java HashMap

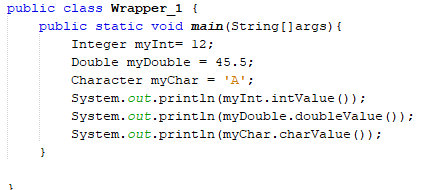


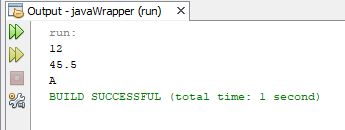
**Hasil :**



* 1. **Hasil Percobaan 14**

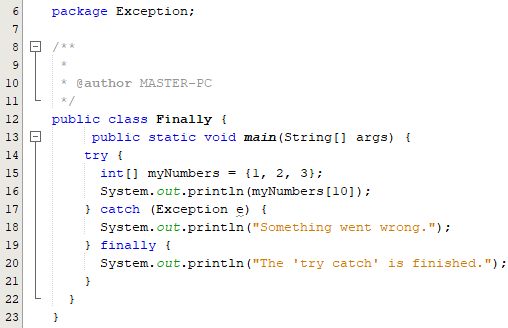
Java Wrapper



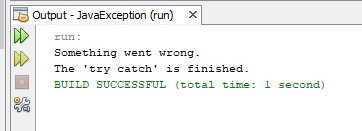


* 1. **Hasil Percobaan 15**

Java Exception



Hasil :



**BAB IV**

**ANALISIS HASIL PERCOBAAN**

* 1. **Analisis hasil percobaan 1**

**Java Constructor**

* Dengan parameter

Pada class myprofile yang termasuk constructor adalah

*public MyProfile(long nim, String name,String alamat,String jurusan,int angkatan) {*

*modelNim = nim;*

*modelName = name;*

*modelAlamat = alamat;*

*modelJurusan = jurusan;*

*modelAngkatan = angkatan;*

*}*

Script di atas merupakan deklarasi konstruktor dengan 5 parameter yaitu nim , name , alamat , jurusan , angkatan .Yang mana parameter tersebut akan di tetapkan ke bidang objek . untuk memanggil constructor kita harus membuat instance objek seperti di bawah :

*MyProfile myProfile = new MyProfile(201869040018L,"Lailatul Afidah","pandaan","Teknik Informatika",2018);*

Untuk meneruskan parameter ke constructor maka sertakan parameter di dalam tanda kurung setelah nama kelas di sisi kanan tanda sama dengan. Objek kemudian dibuat, dan konstruktor dieksekusi. Setelah eksekusi constructor di atas, bidang yang diinisialisasi oleh konstruktor akan memiliki nilai parameter yang diteruskan ke konstruktor

* Tanpa parameter

Constructor tanpa parameter akan menginisialisasi variable instance sesuai dengan type datanya.dan disebut juga dengan constructor default

*public class MyProfile2 {*

*long Nim; // Create a class attribute*

*String Nama;*

*String Alamat;*

*String Jurusan;*

*int Angkatan;*

*public MyProfile2() {*

*Nim = 201869040018L;*

*Nama = "Lailatul Afidah";*

*Alamat = "Pandaan";*

*Jurusan = "Teknik Informatika";*

*Angkatan = 2018;*

*}*

Dari 5 variabel constructor di atas maka akan di eksekusi sesuai dengan tipe datanya

Untuk memangggil nya kita membuat instance objek seperti di bawah :

*MyProfile2 myProfile = new MyProfile2();*

Script di atas untuk memanggil constructor tanpa parameter .

**4.2 Analisis hasil percobaan 2**

**Java modifiers**

* Public

public berarti bahwa semua kode dapat mengakses kelas, attribute , konstruktor atau metode, sesuai dari mana kode mengakses berada. Kode mengakses dapat berada di kelas yang berbeda dan package yang berbeda.

*public class person {*

kelas di atas menggunakan akses modifiers public jadi bisa di akses dari mana saja

*public String fname = "John";*

*public String lname = "Doe";*

*public String email = "john@doe.com";*

*public int age = 24;*

untuk variable *fname, lname ,email,age* kita membuat acces public jadi bisa diakses di kelas lain ,package lain

* Private

Kode hanya dapat diakses di dalam kelas yang dideklarasikan

*public class person1 {*

kelas di atas menggunakan acces modifier public jadi bias di akses dari mana saja

*private String fname = "lailatul ";*

*private String lname = "Afidah";*

*private String email = "tull27493@gmail.com";*

*private int age = 19;*

untuk variable fname,lname,email ,dan age menggunakan acces modifier private jadi hanya bias di akses didalam kelas yang di deklarasikan tidak bias di akses di kelas lain maupun package yang lain

* Default

*class Person {*

jadi kita membuat kelas dengann acces modifier default yang mana Kelas hanya dapat diakses oleh kelas-kelas dalam paket yang sama.

*String fname = "John";*

*String lname = "Doe";*

*String email = "john@doe.com";*

*int age = 24;*

untuk attribute dari setiap variable di atas kita menggunakan acces modifier default yang mana kode tersebut hanya dapat di akses dari package yang sama.

* Protected

*class Person {*

jadi kita membuat kelas dengann acces modifier default yang mana Kelas hanya dapat diakses oleh kelas-kelas dalam paket yang sama.

*protected String fname = "lailatul";*

*protected String lname = "afidah";*

*protected String email = "tull2493@gmail.com";*

*protected int age = 12;*

*}*

Variable attribute di atas menggunakan protected yang mana Kode ini dapat diakses dalam paket dan subclass yang sama.

*class Student extends Person {*

kode di atas menggunakan keyword ekstends untuk mendapatkan attribute dari kelas induk /subclass ke dalam kelas dasar

*private int graduationYear = 2018;*

jadi kelas student berisi attribute *lname,fname email,age ,dan graduationYear*

* 1. **Analisis hasil percobaan 3**

**Java encapsulation**

*public class Person {*

menggunakan public karena bisa di akses dari luar kelas

*private String Name ;*

variable name kita menggunakan private agar tidak bisa dia akses darikelas luar untuk menyembunykan data tersebut

*public String getName(){*

*return Name;*

untuk metode get artinya mendapatkan nilai dari variable . dan ini memilik nilai return

*}*

*//set*

*public void setName(String newName){*

*this.Name = newName;*

Metode Set artinya mengambil parameter (newName) dan mengisi data ke dalam objek ,kata this merupakan kunci yang digunakan untuk merujuk ke objek saat ini.

*}*

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

*Person data = new Person();*

*data.Name = "rara";*

*System.out.println(data.Name);*

*}*

*}*

* 1. **Analisis hasil percobaan 4**

Java Package /API

* Java Scanner

*import java.util.Scanner;*

untuk mengimpor data yang ada pada package java.util.\*

*public class Input\_Biodata {*

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

*Scanner input = new Scanner (System.in);*

berfungsi untuk mencetak hasil dari data-data yang di inputkan

*long nim ;*

*int Thun\_angkatan ;*

*String nama ;*

*String alamat ;*

*String jurusan;*

*String jenis\_kelamin;*

membuat variable untuk menampung nilai yang di inputkan dari user

*System.out.print("Nama :");*

*nama = input.nextLine();*

*System.out.print("ALamat :");*

*alamat = input.nextLine();*

*System.out.print("Jurusan :");*

*jurusan = input.nextLine();*

*System.out.print("Jenis Kelamin :");*

*jenis\_kelamin = input.nextLine();*

*System.out.print("Nim :");*

*nim = input.nextLong();*

*System.out.print("Thun Angkatan :");*

*Thun\_angkatan = input.nextInt();*

Untuk menyimpan data kita sesuai kan dengan tipe data nya jika string seperti di atas makan menggunakan method nektline () atau next() untuk long nextLong()

* Buffer – reader

import java.io.\*;

untuk mendapatkan input dari user

*public class input\_Buffer {*

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

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

Membaca sebuah variable yang akan diinput dari suatu program jadi yang di baca nanty nilai nama , alamat, jurusan , jenis kelamin.

*String nama ;*

*String alamat ;*

*String jurusan;*

*String jenis\_kelamin;*

Statement diatas merupakan tempat untuk menyimpan input dari user.

*try{*

*System.out.print("Nama :");*

untuk memberikan output string pada layar

*nama = input.readLine();*

Ketika menggunakan BufferedReader untuk mendapatkan input dari user diperlukan method readLine().

*System.out.print("ALamat :");*

*alamat = input.readLine();*

*System.out.print("Jurusan :");*

*jurusan = input.readLine();*

*System.out.print("Jenis Kelamin :");*

*jenis\_kelamin = input.readLine();*

Method diatas memanggi input.readLine(), mendapatkan input dari user dan memberikan nilai sebuah String. Nilai tersebut akan disimpan ke dalam variabel yang sudah dideklarasikan, yang akan digunakan pada statement akhir untuk menampilkan nilai dari variable tersebut .:

*System.out.println("Nama = " +nama );*

*System.out.println("Alamat = " +alamat );*

*System.out.println("Jurusan = " +jurusan );*

*System.out.println("Jenis kelamin = " +jenis\_kelamin );*

*}catch(IOException ex){*

*System.out.println("data yang diisi tidak sesuai");*

Pada baris ini menjelaskan bahwa kemungkinan terjadi error pada statement

* JOption\_pane

*import javax.swing.JOptionPane;*

 untuk dapat menggunakan kelas JOptionPane maka kita harus melakukan import ke package javax.swing.

*public class input\_JOptionPane {*

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

*String nama ;*

*String hobi;*

*nama = JOptionPane.showInputDialog("siapa nama kamu ?");*

*hobi = JOptionPane.showInputDialog("Apa hobi mu ");*

Membuat sebuah JOptionPane input dialog, yang akan menampilkan dialog box dengan sebuah pesan, textfield untuk input dan tombol OK seperti gambar diatas. Hasil dari inputan tersebut akan disimpan dalam variabel nama yang bertipe String .

*JOptionPane.showMessageDialog(null,"salam Kenal " + nama);*

*JOptionPane.showMessageDialog(null,"wow keren your hobi is " + hobi);*

  menampilkan sebuah dialog box yang memiliki sebuah pesan dan tombol OK

* Console

Class Console hampir sama dengan BufferReader. Dia juga menggunakan fungsi readLine() untuk mengambil input.

Akan tetapi…Class ini hanya bisa digunakan di lingkungan console saja, seperti Terminal dan CMD.

Class Console tidak bisa digunakan langsung di Netbeans.

Maka dari itu, kita harus kompilasi secara manual.Untuk menggunakan class ini, kita perlu mengimpornya terlebih dahulu.

*import java.io.Console;*

*public class input\_example {*

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

*java.io.Console cnsl = System.console();*

*String nama;*

*int age;*

*System.out.print("Siapa nama kamu :");*

*nama = cnsl.readLine();*

*System.out.print("Berapa umur mu :");*

*age = Integer.parseInt(cnsl.readLine());*

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

*System.out.println("Umur : " + age); }*

* 1. **Analisis hasil percobaan 5**

Java inheritence

*public class Vehicle {*

merupakan super class

*protected String brand = "Ford";*

hanya dapat di akses pada package dan sublclass yang sama

*public void honk(){*

*System.out.println("Tuut........Tuuut!");*

*}*

*}*

*class Car extends Vehicle{*

kelas car merupakan kelas turunan dari kelas vehicle jadi semua data yang ada pada kelas vehicle akan masuk pada kelas car

*private String ModelName = "Mustag";*

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

*Car myCar = new Car();*

*myCar. honk();*

*System.out.println(myCar.brand + " " +myCar.ModelName);*

*}*

*}*

* Study kasus

*import java.util.Scanner;*

*public class Bangun\_datar {*

kelas bangun datar merupakan super class dari banyak kelas

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

*int pil;*

*Lingkaran L = new Lingkaran();*

*Segitiga S = new Segitiga();*

*Persegi P = new Persegi();*

*PersegiPanjang PP = new PersegiPanjang();*

Kelas kelas yang merupakan inherin dari kelas bangun datar .jadi kode di atas untuk mendeklarasikan input an ke dalam java .

*Scanner inp = new Scanner (System.in);*

Untuk mendeklarasika variable inp ke dalam java dari data yang di inputkan

* + Lingkaran

*public class Lingkaran extends Bangun\_datar{*

merupakan inherence dari kelas bangun datar dan menggunakan method acces public karena mengaksesnya dari kelas lain.

*public void Lingkaran(){*

*double ll,rl,kl,phi = 3.14;*

*Scanner b = new Scanner (System.in);*

Untuk mendeklrasikan nilai yang di inputkan pada java pada saat memilih pilihan lingkaran

*System.out.println("Lingkaran");*

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

*System.out.print("Masukkan jari - jari :");*

*rl = b.nextDouble();*

mendapatkan input dari user dan memberikan nilai sebuah Double . maka kita menggunakan method nextDouble();

*ll = phi \* rl \* rl;*

*kl = 2\*phi\*rl;*

* + Persegi

*public class Persegi extends Bangun\_datar{*

merupakan inheritence dari kelas bangun datar dan menggunakan method acces public agar bisa di akses di kelas lain .

*public void persegi(){*

*double sp,lp,kp;*

*Scanner c = new Scanner (System.in);*

Untuk mendeklrasikan nilai yang di inputkan pada java pada saat memilih pilihan persegi

*System.out.println("PERSEGI");*

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

*System.out.print("Masukkan sisi :");*

*sp = c.nextDouble();*

*lp = sp \* sp ;*

*kp = 4 \* sp;*

* + Persegi Panjang

*public class PersegiPanjang extends Bangun\_datar{*

*public void PersegiPanjang(){*

*double pp,lep,lp,kp;*

*Scanner d =new Scanner (System.in);*

*System.out.println("PERSEGI PANJANG");*

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

*System.out.print("Masukkan Panjang :");*

*pp = d.nextDouble();*

*System.out.print("Masukkan Lebar :");*

*lep = d.nextDouble();*

*lp = pp \* lep ;*

*kp = (2\*pp)+(2\*lep);*

* + Segitiga

*public class Segitiga extends Bangun\_datar{*

*public void Segitiga(){*

*double ls,as,ts,ss,ks;*

*Scanner a = new Scanner (System.in);*

*System.out.println("PERSEGI");*

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

*System.out.print("Masukkan Alas :");*

*as = a.nextDouble();*

*System.out.print("Masukkan Tinggi :");*

*ts = a.nextDouble();*

*System.out.print("Masukkan Sisi :");*

*ss = a.nextDouble();*

*ls = (as \* ts)/2;*

*ks = (ss \* 2)+ as;*

untuk penjelasan bangun ruang persegi , persegi Panjang ,dan segitiga sama halnya dengan lingkaran jadi mereke merupakan inheritance dari kelas Bangun Ruang .

* 1. **Analisis hasil percobaan 6**

Java polymorishme

*public class School {*

mendeklarasikan kelas dengan method public agar bisa di akses darai mana saja

*public void SchoolRuler(){*

keyword void di gunakan untuk membuat method yang tidak mengembailikan nilai

*System.out.println("Peraturan untuk siswa dan guru : ");*

*}*

*}*

*class Student extends School{*

kode di atas menggunakan keyword ekstends untuk mendapatkan nilai attribute dari kelas induk /subclass ke dalam kelas dasar.jadi kelas students merupakan turunan dari kela School

*public void SchoolRuler(){*

*System.out.println("Siswa harus mematuhi peraturan di Sekolah");*

*}*

*}*

*class Teacher extends School{*

kode di atas menggunakan keyword ekstends untuk mendapatkan attribute dari kelas induk /subclass ke dalam kelas dasar.jadi kelas students merupakan

*public void SchoolRuler(){*

*System.out.println("guru harus memberikan hak untuk siswa");*

*}*

*}*

*public class Myclass {*

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

*School Schoolruler = new School ();*

*School StudentRuler = new Student();*

*School TeacherRuler = new Teacher();*

Pembuatan objek dari kelas void yang sudah di deklarasikan didalam superclass atau subclass .

*Schoolruler.SchoolRuler();*

*StudentRuler.SchoolRuler();*

*TeacherRuler.SchoolRuler();*

Untuk pemanggilan method dari variable diatas

* 1. **Analisis hasil percobaan 7**

Java Inner Class

* Static

*public class Pekerjaan {*

*private static class kerja{*

*private String Bagian;*

Dari kode di atas kelas dalam dideklarasikan dengan private agar objek luar tidak mengakses kelas dalam . dan kelas dalam kita buat static agar dapat ,memgaksesnya tanpa membuat objek dari kelas luar.

Dari kelas dalam kerja kita mengeksusi nilai variable bagian dengan menggunakan fungsi Get dan set yang mana terdapat keyword this yang menjelaskan bahwa dari calss itu sendiri dan megembalikan nilai bagian dengan keyword return

Untuk membuat objek kita harus membuat objek dari outer class dulu lalu inner class nya seperti di bawah ini

*Pekerjaan.kerja myLanguange = new Pekerjaan.kerja();*

* Non \_Static

Untuk inner class non\_static dalam mengaskesnya membutauhkan objek dari kelas luar dari hasil percobaan dia atas kita mempunyai 2 kelas yang mana masing masingk kelas di deklarasikan dengan modifier private agar tidak dapat di akses oleh kelas lain.

*private class Nilai\_UN{*

*private double Matematika =95 ,IPA = 88 , Bahasa = 85;*

*private void Hasil(){*

*System.out.println("Nilai MTK : " +Matematika);*

*System.out.println("Nilai IPA : " +IPA);*

*System.out.println("Nilai Bahasa : " +Bahasa);*

*}*

*}*

*private class Nama{*

*private String Nama\_Siswa = "Lailatul Afidah";*

*private int No\_Induk = 2200;*

*private void Tampilkan(){*

*System.out.println("Nama : " +Nama\_Siswa);*

*System.out.println("No Induk : " +No\_Induk);*

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

*Siswa MyData = new Siswa();*

Jadi di atas kita membuat objek dari kelas luar dulu setelah itu kita buat objek dari kelas dalam Nilai\_UN dan Nama seperti di bawah agar bisa di eksekusi.

*Siswa.Nilai\_UN data1 = MyData.new Nilai\_UN();*

*Siswa.Nama data2 = MyData.new Nama();*

* 1. **Analisis Hasil percobaan 8**

Java abstraction

*abstract class Manusia {*

keyword abstract di gunakan untuk membuat kelas abstrak .adapun kelas abstract tidak bisa digunakan untuk membuat objek .

*public abstract void ManusiaSound();*

karena kelas di atas merupakan kelas abstrak maka pendeklarasian method menggunakan peubah abstrak . dan tidak memiliki tubuh, tubuh disediakan dari subclass(pewarisan dari).maka kita tidak mungkin membuat kelas abstrak dari kelas manusia.sedangakan untuk menggunakan kelas abstrak maka kita harus mewariskan nya ke kelas yang lain

*public void sleep() {*

*class Cewek extends Manusia {*

*public void ManusiaSound()*

kelas Cewek merupakan turunan / pewaris dari kelas Manusia dengan menggunakan keyword extends. Jadi kelas abtrak akan di eksekusi di kelas ini .

maka yang bisa untuk membuat objek yaitu kelas cewek.

*Cewek myCewek = new Cewek(); // Create a Pig object*

* 1. **Analisis Hasil Percobaan 9**

Java Interface

Interface sama dengan class abstrak hanya dalam mendeklrasikan interface kita tambahkan dengan kata kunci interface sebelum nama interface .

*interface FirstInterface {*

*public void Kewajiban();*

*public void Hak();*

*}*

*interface SecondInterface {*

*public void Ayah();*

*public void Ibu();*

*}*

Dari study kasus di atas kami menggunakan mulitiple interface yang mana pada percobaan di atas kami menggunkan 2 interface yaitu FirstInterface dan SecondInterface .

Dan untuk mengaskses method pada interface maka interface harus di implementasikan oleh kelas lain dengan menggunakan keyword implements . Untuk mengimplementasikan beberapa interface, pisahkan dengan koma

*class DemoClass implements FirstInterface,SecondInterface {*

adapun isi dari metode interface di sediakan oleh kelas implements

*public void Kewajiban() {*

*System.out.println("Setiap Anggota Keluarga Wajib Mematuhi Peraturan ");*

*}*

*public void Hak() {*

*System.out.println("Setipa Anggota Kelurga wajib Mendapatkan Hak nya");*

*}*

*public void Ayah(){*

*System.out.println("Nama Ayah : Muahmmmad Yusuf");*

*}*

*public void Ibu(){*

*System.out.println("Nama Ibu : Husnul Humadiyah");*

* 1. **Analisis Hasil Percobaan 10**

Java Enums

Di dalam java enum hanya berisi konstanta yang tetap . Untuk membuat enum, menggunakan keyword **enum** (bukan kelas atau interface), dan memisahkan konstanta dengan koma. Catatan bahwa mereka harus dalam huruf besar seperti

*enum Hari{*

*SENIN,SELASA,RABU,KAMIS,JUMAT,SABTU,MINGGU,*

*}*

Untuk dapat mengakses ***enum*** konstanta dengan **dot(.)** sintaks :

*Hari MyVar = Hari.SENIN;*

*switch(MyVar){*

*case SENIN://dimana konstanta tersebut akan di cocokkan dengan value*

*System.out.println("==========TODAY IS MONDAY=========");*

*// pernyataan yang akan di kerjakan jika****value*** *cocok dengan salah satu data dari constanta tersebut*

*break;*

*case SELASA:*

*System.out.println("==========TODAY IS TUESDAY===========");*

*break;*

*case RABU:*

*System.out.println("=========TODAY IS WEDNESDAY==========");*

Adapun perintah **Break** adalah perintah untuk mengakhiri **statement**. Apabila tidak ada break, komputer akan mengeksekusi intruksi yang berada di bawahnya walaupun berada di case yang berbeda.

* 1. **Analisis Hasil Percobaan 11**

Java User Input

Untuk java user input sama dengan input menggunakan scanner yang mana kita harus mengimpor terlebih dahulu menggunakan syntak di bawah

*import java.util.Scanner;*

kita buat kelas java yang bernama Mine dengan menggunakan peubah public

*public class Mine {*

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

*Scanner myObj = new Scanner(System.in);*

Untuk mengiputkan data kita buat objek seperti dia atas agar data dapat di kompilasi oleh

*System.out.println("\n My Name : Afidah "*

*+ "\n My age : 19 "*

*+ "\n ipk : 4.00");*

Untuk outputan Data yang akan kita kompilasi ada 3 variable yang mana setiap variable di jelaskan pada script selanjutnya.

*String name = myObj.nextLine();*

Variable name bertipe data String

*int age = myObj.nextInt();*

variable age bertipe data Integer

*double Ipk = myObj.nextDouble();*

variable Ipk bertipe data Double

untuk menjelaskan data yang di inputkan kita gunakan keyword nextLine() untuk data yang bertipe data String ,nextInt untuk data yang bertipe data integer dan nextDouble() untuk data yang bertipe data Double.

* 1. **Analisis Hasil Percobaan 12**

Java Date

Pada program diatas kita menggunakan kelas DateTimeFormatter dengan metode ofPattern() dalam paket yang sama untuk memformat atau menguraikan objek waktu

*import java.time.LocalDateTime;*

kita mengimpor tanggal dan waktu pada program

*import java.time.format.DateTimeFormatter;*

Selanjutnya, mengimpor format tanggal dan waktu yang akan kita atur

Setelah kita mengimpor lalu kita membuat kelas

*public class MyClass {*

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

*LocalDateTime myDateObj = LocalDateTime.now();*

LocalDateTime myObj digunakan Untuk menambahkan objek tanggal dan waktu dengan menggunakan metode Now yang artinya tanggal dan waktu saat ini

*System.out.println("Before formatting: " + myDateObj);*

Ini adalah fungsi untuk menampilkan teks ke layar output-annya dengan menambahkan objek nya.

*DateTimeFormatter myFormatObj =DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");*

Selanjutnya kita mengubah format tanggal dan waktu , dengan format *ofPattern* yang menerima segala macam nilai .

*String formattedDate = myDateObj.format(myFormatObj);*

Setelah itu, menambahkan objek yang akan kita buat dalam memeberikan format yang berbeda pada tanggal dan waktu.

*System.out.println("After Formatting : " + formattedDate);*

Kita memanggil hasil outputan yang dudah di edit pada format yang di ubah.

* 1. **Analisis Hasil Percobaan 13**

Java HashMap

Dalam program HashMap kita membuat sebuah kunci(key) dan Nilai(Value)

*import java.util.HashMap;*

Jadi sebelum membuat kita membuat kelas HashMap kita harus mengimporkan nya terlebih dahulu

*public class Example {*

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

*HashMap<String, Long> people = new HashMap<String, Long>();*

untuk menggunakan HashMap, kita harus membuat objeknya terlebih dahulu. Objek dari Hashmap dapat dibuat dengan kata kunci new.

*people.put("Afidah", 201869040018L);*

*people.put("Laila", 201869040021L);*

*people.put("Angie", 2018690400001L);*

Pada contoh di atas, kita membuat objek HashMap bernama people. Objek ini dapat kita gunakan untuk menyimpan koleksi data.

Tipe data yang digunakan untuk key adalah String dan value adalah Integer.

Artinya: key harus bertipe String dan nilai yang tersimpan harus dalam bertipe Integer dan keyword put untuk mengisi nilai ke objek people.

* 1. **Analisis Hasil Percobaan 14**

Java Wrapper

Dalam program di atas untuk membuat objek wrapper, kita gunakan kelas wrapper alih-alih tipe primitif. Untuk mendapatkan nilai, Anda cukup mencetak objek

*Integer myInt= 12;*

*Double myDouble = 45.5;*

*Character myChar = 'A';*

*System.out.println(myInt.intValue());*

*System.out.println(myDouble.doubleValue());*

*System.out.println(myChar.charValue());*

Jadi di atas kita mmebuat methode sesuai dengan nilai pembungkus nya seperti *intValue(),doubleVslue(),charValue().*

* 1. **Analisis Hasil Percobaan 15**

Java Exception

Untuk program Exception di eksekusi ketika ada kesalahan dalam data suatu program baik data yang tidak valid ataupun karena kesalahan dalam mengkompile .

Program diatas menggunakan pernyatan try dan catch

*public class Finally {*

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

*try {*

pernyataan try untuk menentukan blok kode yang akan diuji untuk kesalahan saat sedang dieksekusi.

*int[] myNumbers = {1, 2, 3};*

*System.out.println(myNumbers[10]);*

*} catch (Exception e) {*

Pernyataan catch  untuk menentukan blok kode yang akan dieksekusi, jika kesalahan terjadi di blok coba.

*System.out.println("Something went wrong.");*

*} finally {*

*System.out.println("The 'try catch' is finished.");*

Jadi program diatas menyatkan bahwa sebuah angka terdiiri dari data int 1,2,3 tetapi karena data dinyatakan angka 10 maka kompilator meminta program menggunakan pengecualian .

**BAB V**

**PENUTUP**

1. **KESIMPULAN**

Pemrograman Berorientasi Objek (*Object Oriented Programming* atau OOP) merupakan paradigma 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**
2. 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)