

**LAPORAN RESMI**  
**MODUL I**  
**CLASS DAN OBJECT**  
**PEMROGRAMAN BERBASIS OBJEK**



<b>NAMA</b>	<b>: WARDATUS SHOLICHA</b>
<b>N.R.P</b>	<b>: 230441100170</b>
<b>DOSEN</b>	<b>: ACHMAD ZAIN NUR S.Kom., M.T</b>
<b>ASISTEN</b>	<b>: DEVI DWI NOVITASARI</b>
<b>TGL PRAKTIKUM</b>	<b>: 23 MARET 2024</b>

**Disetujui : 27 MARET 2024**  
**Asisten**

**DEVI DWI NOVITASARI**  
**22.04.411.00090**



**LABORATORIUM BISNIS INTELIJEN SISTEM**  
**PRODI SISTEM INFORMASI**  
**JURUSAN TEKNIK INFORMATIKA**  
**FAKULTAS TEKNIK**  
**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Java adalah bahasa pemrograman yang populer, dibuat pada tahun 1995. Java digunakan untuk mengembangkan aplikasi seluler, aplikasi web, aplikasi desktop, game, dan banyak lagi. Java bekerja pada platform yang berbeda (Windows, Mac, Linux, Raspberry Pi.) Java juga adalah salah satu bahasa pemrograman paling populer di dunia, Mudah dipelajari dan mudah digunakan. Ini open-source dan gratis. Ini aman, cepat, dan kuat. Ini memiliki dukungan komunitas yang sangat besar. Java adalah bahasa berorientasi objek yang memberikan struktur yang jelas untuk program dan memungkinkan kode untuk digunakan kembali, menurunkan biaya pengembangan. Karena Java dekat dengan C++ dan C# , memudahkan programmer untuk beralih ke Java atau sebaliknya.

Class diumpamakan seperti cetakan yang berguna untuk mencetak suatu object, class juga merupakan grup suatu object dengan kemiripan atribut, behaviour dan relasi ke object lain. Contoh dari sebuah class tersebut misalnya cetakan dari sebuah mobil sedan. Mobil sedan tentunya mempunyai konsep, dan konsep inilah dipakai untuk membentuk/mencetak hingga menjadi mobil sedan. Oleh karena itu konsep bisa diartikan sebagai class.

Object adalah instance dari class. Jika class secara umum merepresentasikan sebuah object, sebuah instance adalah representasi nyata dari class itu sendiri. Contoh : Dari class Kendaraan, maka kita dapat membuat object sepeda, motor, Mobil

### **1.2 Tujuan**

- Mahasiswa mampu memahami bahasa pemrograman Java sehingga dapat mengembangkan sebuah aplikasi.
- Mahasiswa mampu memahami konsep Class dan Object dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya.
- Mahasiswa mampu memahami atribut dan method dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya.

## **BAB II**

### **DASAR TEORI**

#### **2.1 Pengertian Class dan Objek**

Class diumpamakan seperti cetakan yang berguna untuk mencetak suatu object, class juga merupakan grup suatu object dengan kemiripan attribut, behaviour dan relasi ke object lain. Contoh dari sebuah class tersebut misalnya cetakan dari sebuah mobil sedan. Mobil sedan tentunya mempunyai konsep, dan konsep inilah dipakai untuk membentuk/mencetak hingga menjadi mobil sedan. Oleh karena itu konsep bisa diartikan sebagai class.

Object adalah instance dari class. Jika class secara umum merepresentasikan sebuah object, sebuah instance adalah representasi nyata dari class itu sendiri. Contoh : Dari class Kendaraan, maka kita dapat membuat object sepeda, motor, mobil, becak dll.

Pada dasarnya ada dua karakteristik yang utama pada sebuah object yaitu :

1. Setiap object memiliki attribut sebagai status yang akan disebut state
2. Setiap object memiliki tingkah laku yang kemudian akan disebut dengan method (behaviour).

<b>State</b>	<b>Behaviour</b>
Pedal	Kecepatannya menaik
Roda	Kecepatannya menurun
Jeruji	Perpindahan gigi sepeda

Object sepeda

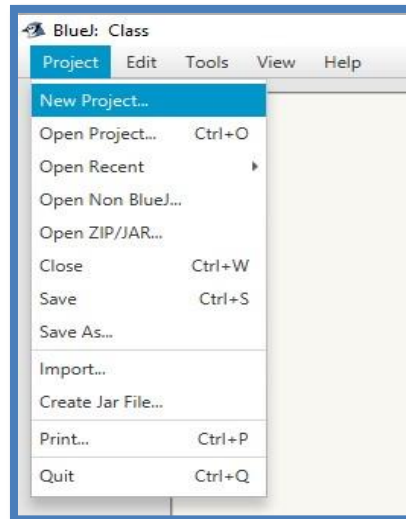
Dalam pengembangan perangkat lunak berorientasi object, object dalam perangkat lunak akan menyimpan state-nya dalam variable dan menyimpan informasi tingkah laku (behaviour) dalam method-method atau fungsi-fungsi. Untuk membuat object, kita menggunakan perintah new dengan sebuah nama class yang akan dibuat sebagai instance dari class.

Contoh :

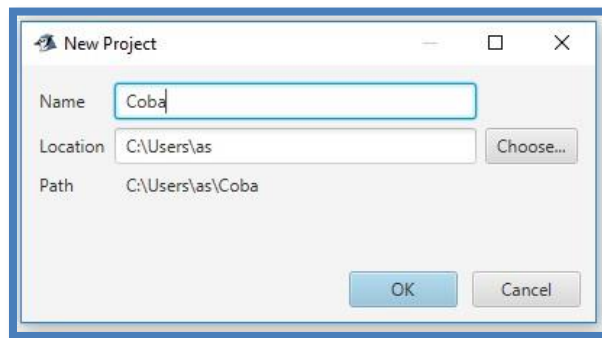
## 1. Tahapan membuat suatu class

Untuk membuat suatu class ada beberapa tahap yang harus di lakukan seperti yang ada pada Gambar.1 dengan rincian tahapan sebagai berikut :

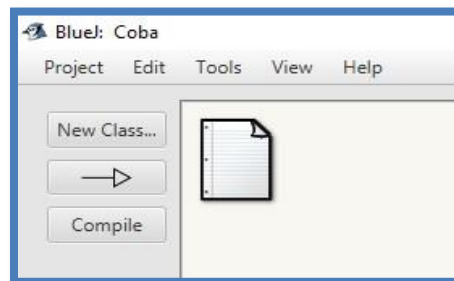
### 2. Buka aplikasi BlueJ, klik project pilih “New Project”.



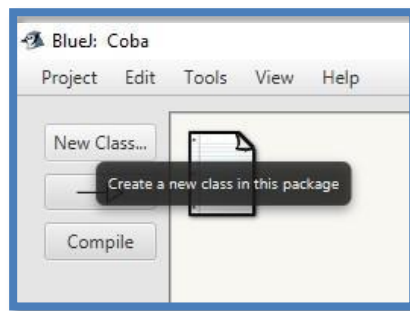
### 3. Pada form new project isilah nama project yang akan kita buat.



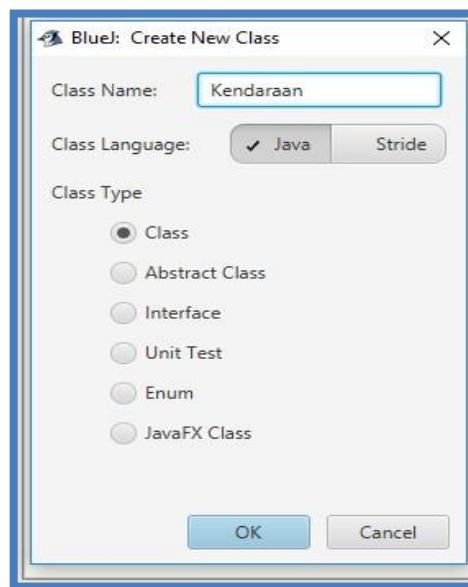
a. Tampilan form setelah nama project diisi.



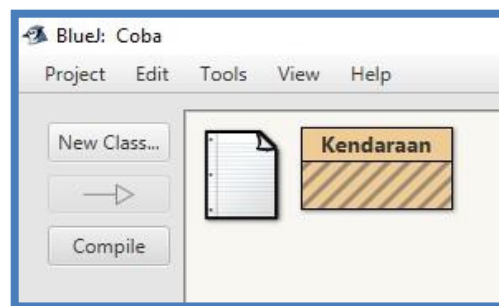
b. Klik “New *Class*”, untuk membuat *class*



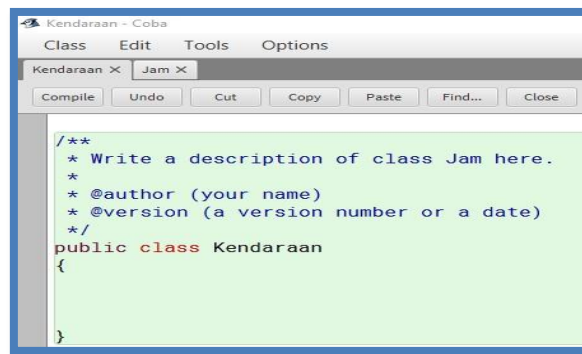
- c. Akan ada form baru yang digunakan untuk mengisi nama *Class* sesuai dengan yang kita inginkan pada kolom “**Class Name**”, lalu tekan “OK”.



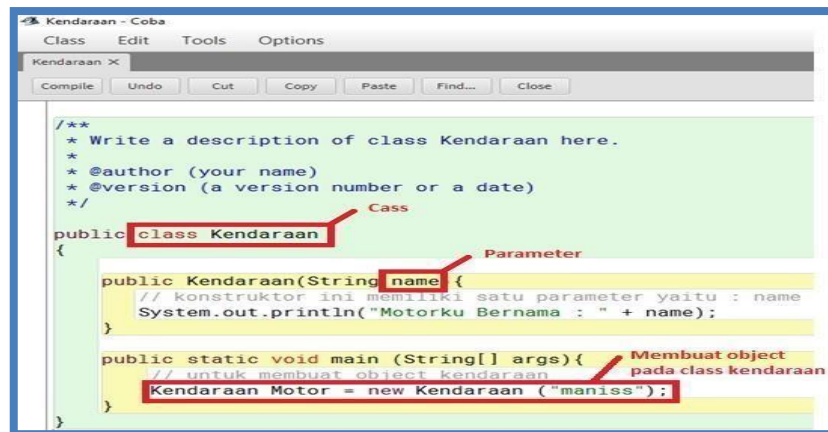
- d. Tampilan form setelah kita membuat *class*, dan kita bisa mengedit code di dalam *class*, pada contoh diatas terdapat pada *class* kendaraan.



- e. Tampilan *class* yang didalamnya bisa kita edit sesuai dengan yang kita inginkan.



#### 4. Contoh membuat *object* dari suatu *class*



Gambar 2. Source code

Pada Gambar.2 merupakan contoh dari suatu *class* Kendaraan yang didalamnya terdapat suatu *object*. Berdasarkan program diatas, telah dibuat *object* Motor dari *class* kendaraan, dan kita juga bisa menggunakan konstruktor yang akan dijalankan secara otomatis pada saat *object* dibuat yakni ketika perintah “new” dijalankan.

Contoh Konstruktor seperti berikut :

```

public Kendaraan (String name){
    // Konstruktor ini memiliki satu parameter yaitu : nama
    System.out.println ("Motorku bernama : " + name);
}

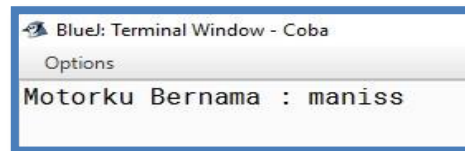
```

Source code diatas disebut dengan konstruktor yang nantinya akan dipanggil secara otomatis pada saat *object* dibuat, tanpa membuat *method* terlebih dahulu. Konstruktor harus menggunakan nama *class* untuk penamaannya, dan dalam suatu *class* hanya boleh ada satu konstruktor, jika tidak maka akan terjadi *SyntaxError*.

Contoh *object* seperti berikut :

```
Kendaraan Motor = new Kendaraan ("maniss");
```

Kode program diatas artinya membuat suatu *object* **Motor** dari *class* **Kendaraan**, dengan menggunakan **keyword new**. Karena menggunakan konstruktor maka tidak perlu menggunakan *method* untuk menginisialisasi *object*.



Gambar 3. Output dari Class Kendaraan

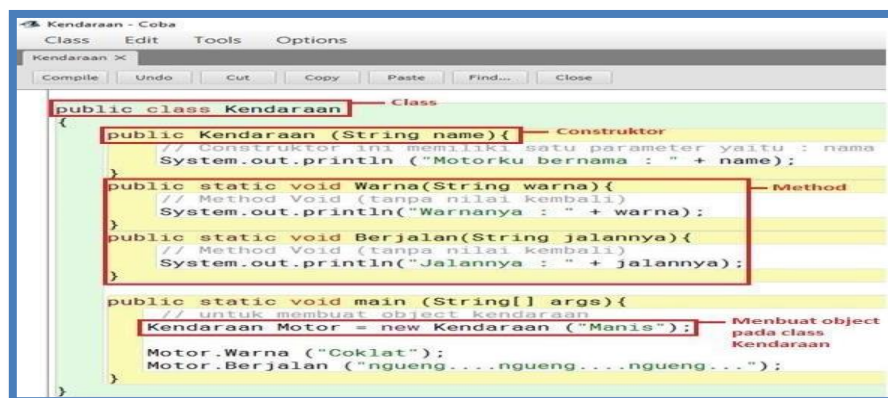
### 2.1.1 Method

**Method** merupakan kumpulan dari fungsi yang kita buat dalam suatu *class*, yang kemudian fungsi-fungsi itu akan dipanggil kembali. dengan *method* kita bisa mempersingkat coding yang kita buat. *Method* didefinisikan pada *class* tetapi akan dipanggil melalui *object*, *method* menentukan perilaku *object*, yakni apa yang akan terjadi saat *object* itu dibuat serta berbagai operasi yang dapat dilakukan *object*.

Struktur dari *method* diantaranya sebagai berikut :

1. *Method* terdiri dari statement public, private, protected, yang menandakan hak akses *method* tersebut.
2. Nama *method* terdiri dari alfabeth saja.
3. Parameter, dalam penulisannya harus diawali dengan tipe data baru nama parameternya.
4. Isi *method*.

Contoh :



Gambar 4

Penjelasan Gambar.1 :

- a. Pada *class* Kendaraan terdapat Konstruktor, *Method*, dan *Object*
- b. Konstruktor merupakan suatu *method* yang akan memberikan nilai awal pada saat suatu *object* dibuat. Jadi apabila *object* diatas telah dibuat lalu dijalankan maka konstruktor akan bekerja secara otomatis. Biasanya nama konstruktor sama dengan nama *Class* yang dibuat.
- c. Pada *method* diatas terdapat (“public static”) yang merupakan modifier yaitu jenis akses *method* yang bersifat public. Pada gambar diatas ada dua *method* yang nantinya akan dipanggil saat *object* dijalankan yaitu *method* Warna dan *Method* Berjalan.
- d. Kode program dibawah artinya : membuat sebuah *object* Motor dari *class* Kendaraan dengan menggunakan keyword new.

```
Kendaraan Motor = new Kendaraan ("Manis");
```

- e. Kode program dibawah artinya : menginisialisasi *object* Motor menggunakan *method* Warna dan *method* Berjalan yang dimiliki *class* kendaraan yang nantinya akan dipanggil oleh *object*.

```
Motor.Warna ("Coklat");  
Motor.Berjalan ("ngueng....ngueng....ngueng...");
```

### 2.1.2 Atribut

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas, Atribut dapat memiliki hak akses private, public maupun protected. Sebuah atribut yang dinyatakan sebagai private hanya dapat diakses secara langsung oleh kelas yang membungkusnya, sedangkan kelas lainnya tidak dapat mengakses atribut ini secara langsung.

Sebuah atribut yang dinyatakan sebagai public dapat diakses secara langsung oleh kelas lain di luar kelas yang membungkusnya. Sebuah atribut yang dinyatakan sebagai protected tidak dapat diakses secara langsung oleh kelas lain di luar kelas yang membungkusnya, kecuali kelas yang mengaksesnya adalah kelas turunan dari kelas yang membungkusnya.

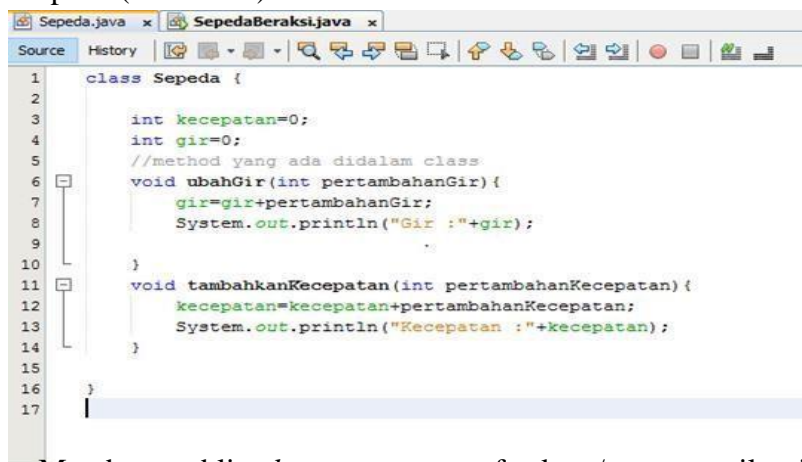


## Karakteristik Atribut

Dalam *class*, atribut disebut sebagai variabel. Atribut dapat membedakan antara satu *object* dengan *object* yang lain. Sebagai contoh, pada *class*: mahasiswa, terdapat *object* mahasiswa si A, dan *object* mahasiswa si B. Yang membedakan antara *object* si A dan si B adalah NPM-nya (Nomor Pokok Mahasiswa) yang merupakan atribut dari *object* tersebut. Pada atribut, terdapat pula dua istilah variabel, yaitu Instance Variabel dan *Class Variable*.

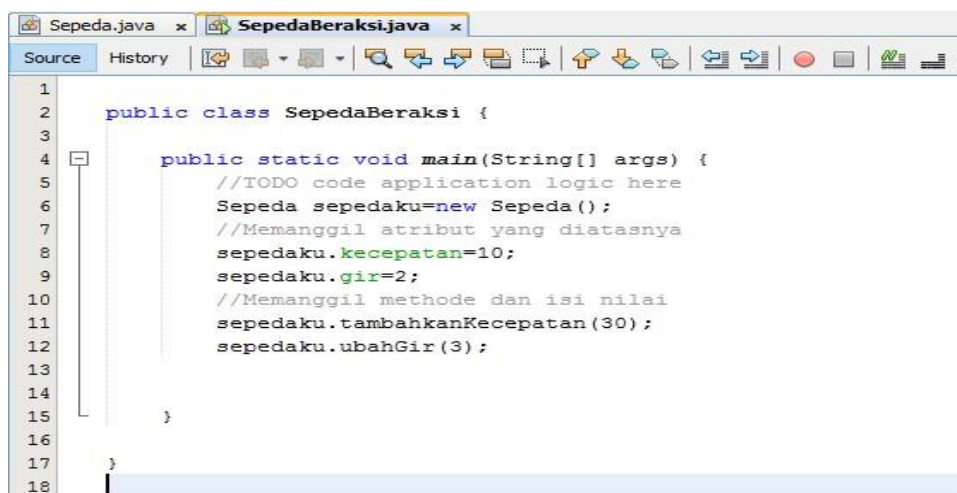
## Contoh penerapan atribut

Membuat *class* dengan nama sepeda dan semua atribut yang dimiliki oleh kelas sepeda (Gambar 1).



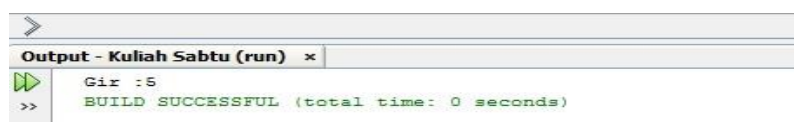
```
1 class Sepeda {
2
3     int kecepatan=0;
4     int gir=0;
5     //method yang ada didalam class
6     void ubahGir(int pertambahanGir){
7         gir=gir+pertambahanGir;
8         System.out.println("Gir :"+gir);
9     }
10
11     void tambahkanKecepatan(int pertambahanKecepatan){
12         kecepatan=kecepatan+pertambahanKecepatan;
13         System.out.println("Kecepatan :"+kecepatan);
14     }
15 }
16
17
```

Membuat *public class* yang memanfaatkan / memanggil atribut dalam kelas lainnya (Gambar 2).



```
1
2 public class SepedaBeraksi {
3
4     public static void main(String[] args) {
5         //TODO code application logic here
6         Sepeda sepedaku=new Sepeda();
7         //Memanggil atribut yang diatasnya
8         sepedaku.kecepatan=10;
9         sepedaku.gir=2;
10        //Memanggil methode dan isi nilai
11        sepedaku.tambahkanKecepatan(30);
12        sepedaku.ubahGir(3);
13
14    }
15 }
16
17
18
```

Hasil program setelah dijalankan (Gambar 3)



```
>
Output - Kuliah Sabtu (run) x
Gir : 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

## BAB III

### Tugas Pendahuluan

#### 3.1 Soal

1. Jelaskan perbedaan class dan objek dengan menggunakan bahasamu sendiri !

#### 3.2 Jawaban

- 1 A. Class merupakan konsep dasar dalam pemrograman berbasis objek yang berfungsi sebagai blueprint atau cetak biru untuk membuat objek. class akan menentukan apa yang dimiliki sebuah objek (atribut) dan apa yang dapat dilakukan objek (metode).

Contoh : Manusia, hewan, Kendaraan.

- B. Objek merupakan bentuk nyata / real / hidup dari sebuah class yang digambarkan dengan atribut dan method.

## **BAB IV**

### **IMPLEMENTASI**

#### **4.1 Source Code**

1. Program class Mansuia

a. Source code Main class

```
package manusia;
```

```
public class Manusia {  
    public static void main(String[] args) {  
        method identitas = new method();  
        identitas.nama="Wardatus Sholicha";  
        identitas.umur=19;  
        identitas.alamat="Surabaya";  
  
        System.out.println("Nama Saya "+identitas.nama);  
        System.out.println("Umur Saya "+identitas.umur);  
        System.out.println("Alamat Saya "+identitas.alamat);  
  
        identitas.berjalan();  
        identitas.berlari();  
    }  
}
```

b. Source code class atribut&method

```
public class method {  
    String nama;  
    int umur;  
    String alamat;  
  
    void berjalan() {  
        System.out.println("Saya sedang berjalan-jalan di taman");  
    }  
}
```

```

        void berlari() {
            System.out.println("Saya sedang berlari-lari kecil");
        }
    }
}

```

## 2. Program Class Mahasiswa

```

package mahasiswa;
import java.util.Scanner;
public class Mahasiswa {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String nama,nim,prodi,alamat;

        System.out.print("Masukan Nama : ");
        nama= sc.nextLine();
        System.out.print("Masukan Nim : ");
        nim= sc.nextLine();
        System.out.print("Masukan Prodi : ");
        prodi= sc.nextLine();
        System.out.print("Masukan Alamat : ");
        alamat= sc.nextLine();
        System.out.println("");
        System.out.println("");
        System.out.println("=====");
        System.out.println("  Data Mahasiswa  ");
        System.out.println("=====");
        System.out.println("Nama   : " + nama);
        System.out.println("NIM    : " + nim);
        System.out.println("Prodi  : " + prodi);
        System.out.println("Alamat : " + alamat);
    }
}

```

## 4.2 Penjelasan

### 1. Penjelasan program class Manusia

- Program ini terdiri dari dua class : Class Manusia dan class method.

Dimana class ini memiliki tiga atribut:

nama: Untuk menyimpan nama manusia.

umur: Untuk menyimpan umur manusia.

alamat: Untuk menyimpan alamat manusia.

- Selain itu, class method memiliki dua metode yaitu,
  - berjalan(): Metode ini mencetak pesan ke konsol yang menyatakan bahwa orang tersebut sedang berjalan.
  - berlari(): Metode ini mencetak pesan ke konsol yang menyatakan bahwa orang tersebut sedang berlari.
- Class Manusia Ini adalah kelas utama dari program, yang mengandung metode main titik masuk eksekusi untuk program Java. Di dalam metode main, objek dari class method diciptakan dan inisialisasi dilakukan untuk atribut nama, umur, dan alamat dengan nilai tertentu ("Wardatus Sholicha", 19, dan "Surabaya").
- Setelah atribut diinisialisasi, program mencetak nilai atribut tersebut ke konsol untuk menampilkan informasi tentang objek. Kemudian, program memanggil metode berjalan() dan berlari() pada objek tersebut, yang menghasilkan pesan di konsol tentang aktivitas yang dilakukan oleh "Wardatus Sholicha".

### 2. Penjelasan Program Class Mahasiswa

Program tersebut merupakan sebuah program sederhana dalam bahasa pemrograman Java yang bertujuan untuk meminta input dari pengguna berupa nama, NIM (Nomor Induk Mahasiswa), program studi, dan alamat mahasiswa. Setelah menerima input, program akan menampilkan data mahasiswa yang telah dimasukkan oleh pengguna. Program ini menggunakan kelas Scanner untuk membaca input dari pengguna melalui keyboard. Setelah semua input diperoleh, program akan mencetak data mahasiswa ke layar dalam format yang ditentukan.

## 4.3 Hasil

### 1. Hasil dari program class Manusia

```
Output - Manusia (run)

run:
Nama Saya Wardatus Sholicha
Umur Saya 19
Alamat Saya Surabaya
Saya sedang berjalan-jalan di taman
Saya sedang berlari-lari kecil
BUILD SUCCESSFUL (total time: 0 seconds)
```

### 2. Hasil dari program class Mahasiswa

```
Output - Mahasiswa (run)

run:
Masukan Nama : Wardatus Sholicha
Masukan Nim : 230441100170
Masukan Prodi : Sistem Informasi
Masukan Alamat : Surabaya

=====
      Data Mahasiswa
=====
Nama      : Wardatus Sholicha
NIM       : 230441100170
Prodi     : Sistem Informasi
Alamat    : Surabaya
BUILD SUCCESSFUL (total time: 26 seconds)
```

## **BAB V**

### **PENUTUP**

#### **5.1 Analisa**

Dari hasil pratikum, pratikan menganalisa bahwa Java adalah bahasa pemrograman berorientasi objek. Segala sesuatu di Java dikaitkan dengan kelas dan objek, bersama dengan atribut dan metodenya. Sebagai contoh: dalam kehidupan nyata, mobil adalah sebuah objek. Mobil memiliki atribut , seperti berat dan warna, dan metode , seperti drive dan rem. Kelas seperti konstruktor objek, atau "cetak biru" untuk membuat objek.

Class adalah cetak biru/prototipe/pendefinisian dari suatu benda. Didalam class-lah atribut dan method suatu object didefinisikan. class juga merupakan grup suatu object dengan kemiripan atribut, brhaviour dan relasi ke object lain Contoh: Manusia, Window.

Object adalah bentuk instance/nyata/real/hidup dari sebuah class. Jika class secara umum merepresentasikan sebuah object, sebuah instance adalah representasi nyata dari class itu sendiri. Contoh : Shelly:Manusia (Object Shelly mempunyai Class Manusia), Form1:Window (Object Form1 mempunyai class Window). Di Java, sebuah objek dibuat dari sebuah kelas.

#### **5.2 Kesimpulan**

1. Class diumpamakan seperti cetakan yang berguna untuk mencetak suatu object, class juga merupakan grup suatu object dengan kemiripan atribut, brhaviour dan relasi ke object lain.
2. Object adalah instance dari class. Jika class secara umum merepresentasikan sebuah object, sebuah instance adalah representasi nyata dari class itu sendiri.
3. Setiap object memiliki atribut sebagai status yang akan disebut state.
4. Setiap object memiliki tingkah laku yang kemudian akan disebut dengan method (behaviour).