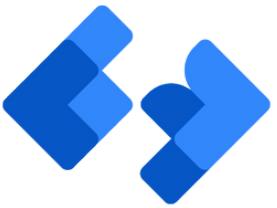


Bab 1 :

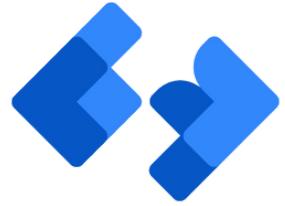
Pengenalan Pemrograman Berorientasi Object



Pengertian OOP

OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berbasis kepada objek. OOP mengadopsi model yang mirip dengan objek-objek di kehidupan sehari-hari. sebuah objek dalam program dapat terdiri dari beberapa atribut dan metode yang saling berhubungan dan berinteraksi, hal tersebut memungkinkan attribut dan method memiliki peran dan fungsinya sendiri dalam membentuk suatu objek secara keseluruhan.





Class

Class merupakan rancangan dari sebuah objek yang mendefenisikan atribut (ciri/variabel) dan method (perilaku) umum dari suatu objek yang dibuat . Dalam OOP, sebuah program yang berjalan wajib untuk mendefinisikan class terlebih dahulu, sebelum dapat membuat objek atau melakukan operasi lainnya, karena Class merupakan struktur dasar yang digunakan untuk menciptakan objek-objek dalam OOP.

Class Mobil



Attribute

- tipe
- roda
- kecepatan

Method

- melaju()
- klakson()
- berbelok()



Declarasi class



```
1 class NamaClass{  
2     //deklarasi atribut  
3     //deklarasi method  
4 }
```

- **class** adalah kata kunci yang digunakan untuk mendeklarasikan suatu kelas.
- **NamaClass** merupakan identifier.
- deklarasi tribut dan deklarasi Method bisa berjumlah 1 atau lebih.



Object class dan Drive class

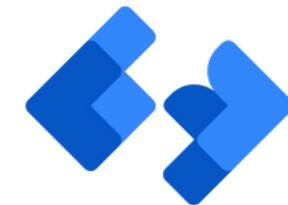
Berdasarkan peran atau karakteristik, class terbagi menjadi 2 jenis:

- **Object Class** : Class yang tidak ada main, biasanya berisi method dan attribute yang nantinya akan di panggil di Drive Class.
- **Drive Class** : Class yang menggunakan main atau dapat di running.

```
● ● ●  
1 public class HelloWorld {  
2     //attribut  
3     String nama;  
4     String pekerjaan;  
5  
6     //method  
7     void pekerjaan() {  
8         System.out.println("nama saya " + nama +  
9                 ",saya seorang" + pekerjaan);  
10    }  
11 }  
12 }
```

```
● ● ●  
1 public class main {  
2  
3     public static void main(String[] args) {  
4         //membuat object  
5         HelloWorld helloworld = new HelloWorld();  
6  
7         //inisialisasi attribut  
8         helloworld.nama = "Firman";  
9         helloworld.pekerjaan = "Mahasiswa";  
10    }  
11  
12    //hasil  
13    helloworld.pekerjaan();  
14 }
```

Object



Object adalah realisasi dari class. Dapat dibayangkan, class adalah sebuah cetakan/template , dan object adalah bentuk dari representasi cetakan/template dari class. Setiap object akan mempunyai state / keadaan (instance variabel/properties) yang membedakan satu object dengan object lain. Terdapat istilah “*instansiasi*” dalam OOP yaitu proses pembuatan object real dari class. Ketika kita membuat sebuah class bukan berarti kita membuat sebuah objek. Ciri-ciri pembuatan objek adalah dengan adanya operator “**new**”.

Class Mobil



Attribute

- tipe
- roda
- kecepatan

Method

- melaju()
- klakson()
- berbelok()

Object VIOS



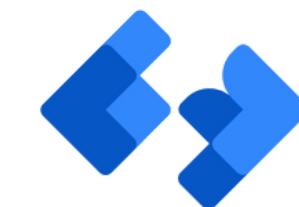
Attribut

- tipe -> sedan
- roda -> 4
- kecepatan -> 200km/jam

Method()

- melaju()
- klakson()
- berbelok()

Deklarasi Object sebagai berikut:



```
1 NamaClass newObject = new NamaClass();
```

NamaClass : nama object class yang sudah kita buat sebelumnya

new : keyword perintah untuk membuat sebuah object

NewObject : nama object class akan kita buat

NamaClass() : constructor dari NamaClass

Attribut



Atribut adalah karakteristik unik atau ciri dari sebuah objek. Karakteristik tersebut dapat berupa data/variabel yang akan dimiliki oleh objek dari kelas tersebut. Atribut dapat memiliki hak akses private, public maupun protected (akan dijelaskan di bab 4).

Attribute dideklarasikan sebagai berikut:

Hak_akses tipe_data nama_Atribut;

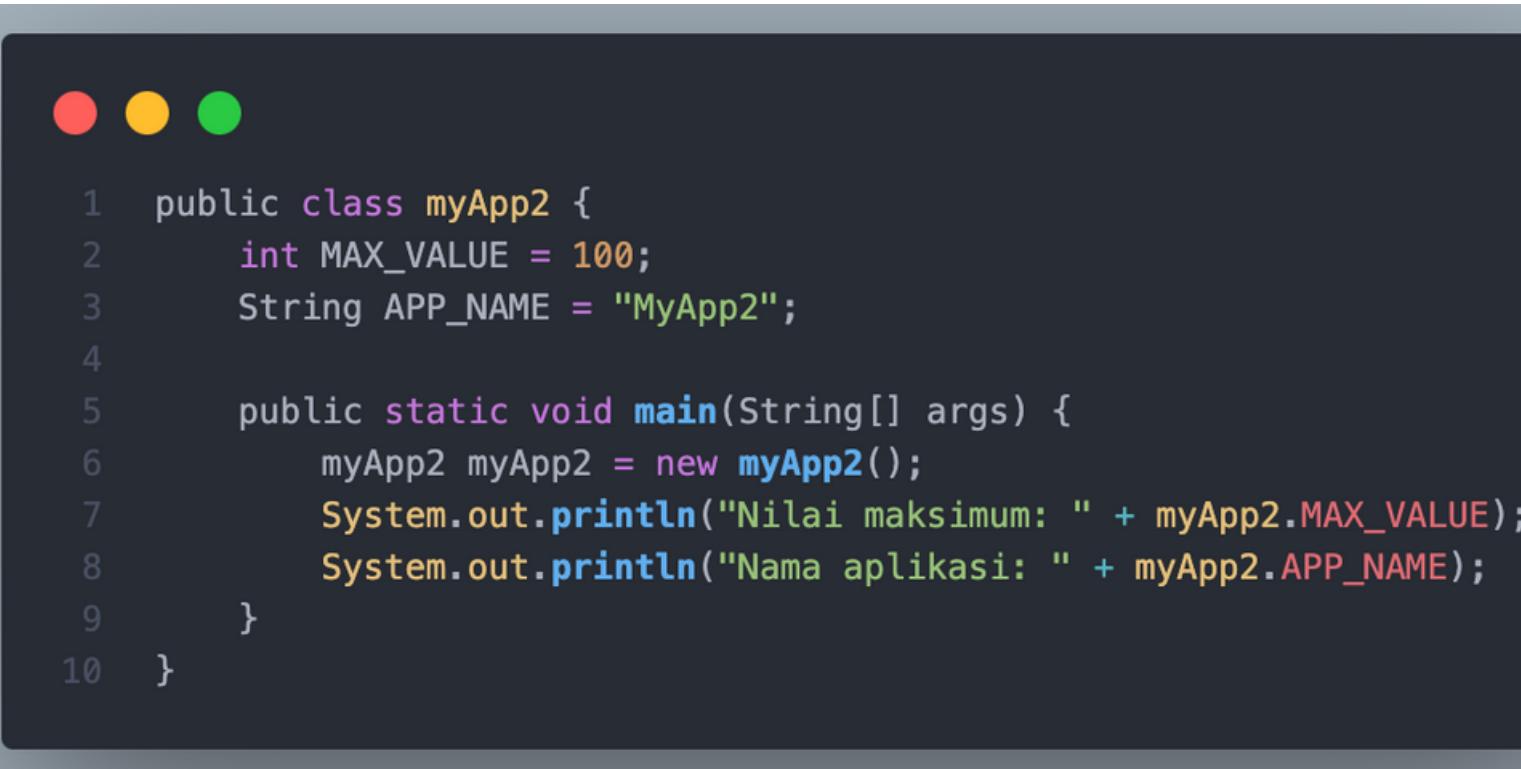
Contoh :

public string merk_kendaraan;

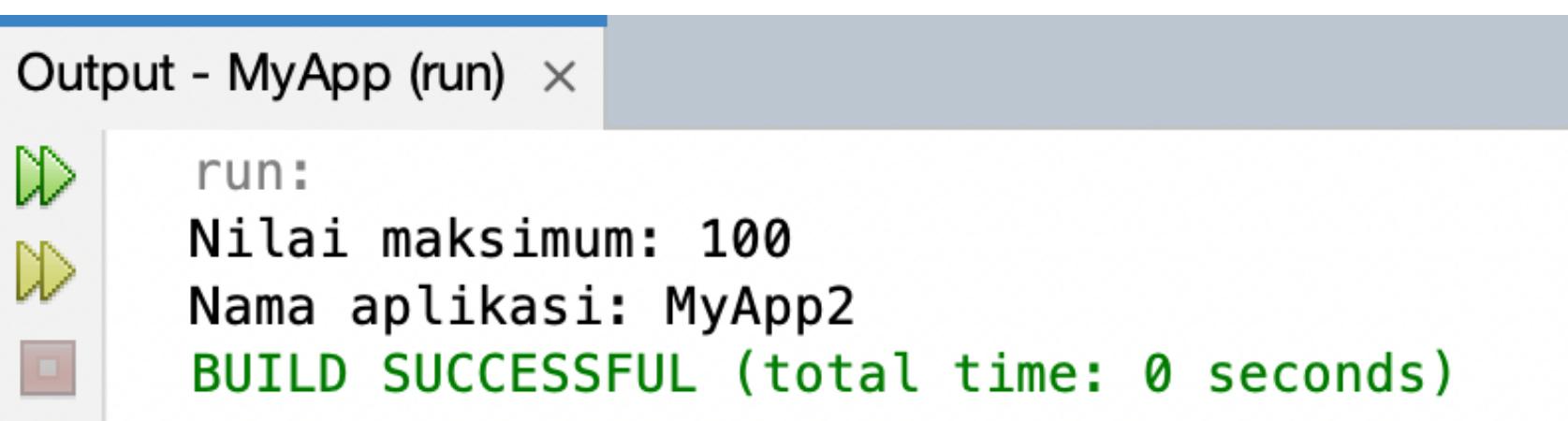
Jenis atribut umum dalam OOP Java:

A. Atribut Instance (*Instance Variables*)

Atribut yang berkaitan dengan objek yang dihasilkan dari class yang dipakai. Atribut ini mendefinisikan karakteristik atau properti unik dari objek.



```
1 public class myApp2 {  
2     int MAX_VALUE = 100;  
3     String APP_NAME = "MyApp2";  
4  
5     public static void main(String[] args) {  
6         myApp2 myApp2 = new myApp2();  
7         System.out.println("Nilai maksimum: " + myApp2.MAX_VALUE);  
8         System.out.println("Nama aplikasi: " + myApp2.APP_NAME);  
9     }  
10 }
```



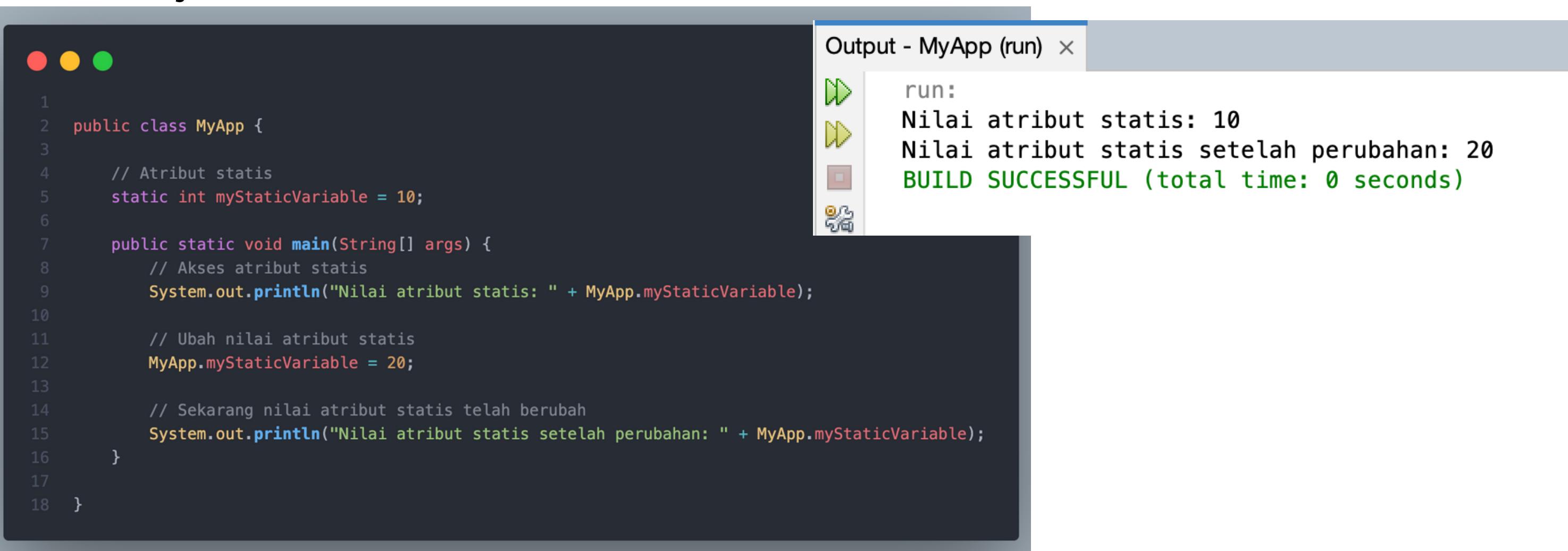
Output - MyApp (run) ×

```
run:  
Nilai maksimum: 100  
Nama aplikasi: MyApp2  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jenis atribut umum dalam OOP Java:

B. Atribut Static (*Static Variables*)

Atribut yang berkaitan dengan class itu sendiri, bukan dengan objek. Biasanya digunakan untuk menyimpan data yang bersifat bersama antara semua objek kelas.



The image shows a Java development environment with a code editor and an output window. The code editor contains the following Java code:

```
1 public class MyApp {
2     // Atribut statis
3     static int myStaticVariable = 10;
4
5     public static void main(String[] args) {
6         // Akses atribut statis
7         System.out.println("Nilai atribut statis: " + MyApp.myStaticVariable);
8
9         // Ubah nilai atribut statis
10        MyApp.myStaticVariable = 20;
11
12        // Sekarang nilai atribut statis telah berubah
13        System.out.println("Nilai atribut statis setelah perubahan: " + MyApp.myStaticVariable);
14    }
15
16 }
17
18 }
```

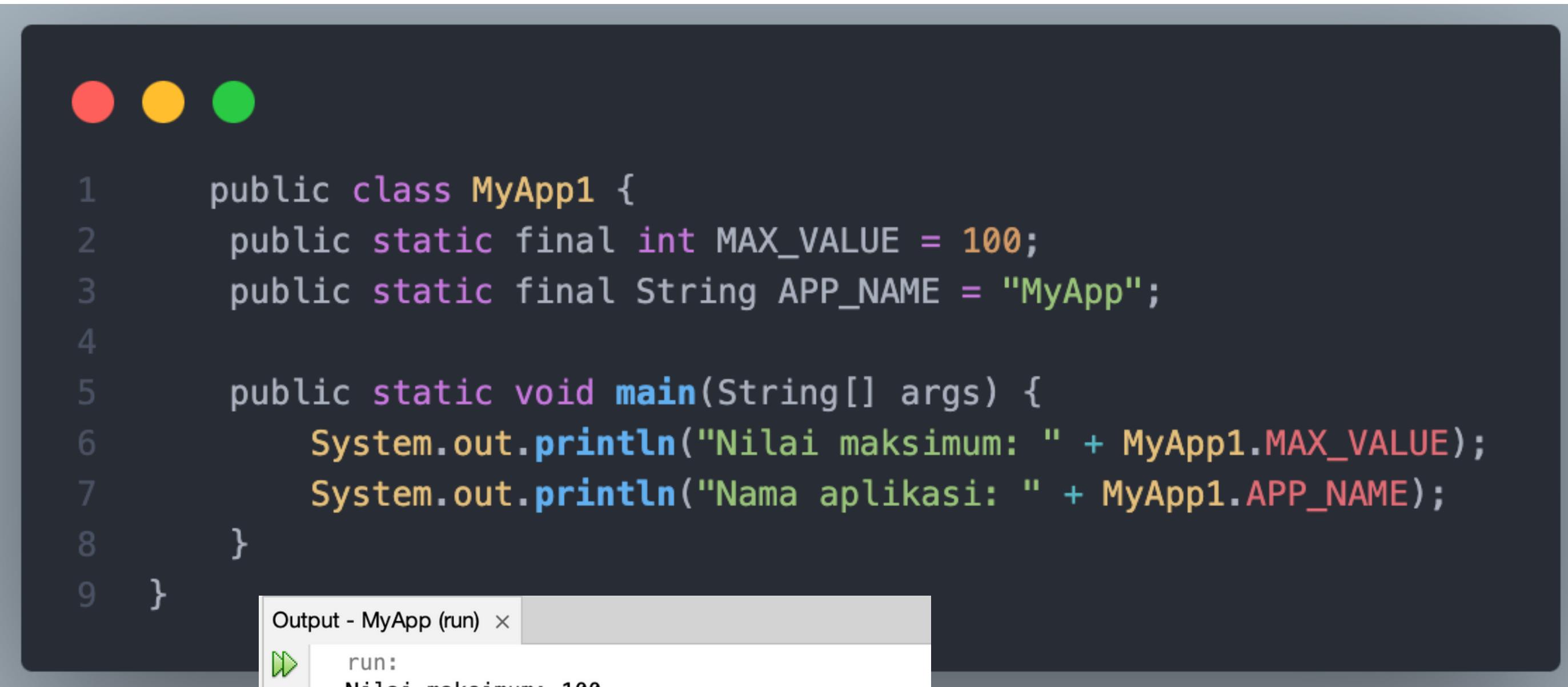
The output window titled "Output - MyApp (run)" shows the following results:

```
run:
Nilai atribut statis: 10
Nilai atribut statis setelah perubahan: 20
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jenis atribut umum dalam OOP Java:

C. Atribut Final (*Final Variables*)

Atribut yang nilai awalnya harus ditetapkan saat deklarasi dan tidak dapat diubah lagi setelahnya. Atribut ini digunakan untuk menyimpan konstanta atau nilai tetap yang tidak boleh diubah.



The image shows a Java application window with three colored circles (red, yellow, green) at the top. Below them is the Java code for `MyApp1`. The code defines a class `MyApp1` with two static final variables: `MAX_VALUE` (100) and `APP_NAME` ("MyApp"). It also contains a `main` method that prints these values to the console. At the bottom, there is an "Output - MyApp (run)" tab showing the execution results: "Nilai maksimum: 100" and "Nama aplikasi: MyApp".

```
1 public class MyApp1 {  
2     public static final int MAX_VALUE = 100;  
3     public static final String APP_NAME = "MyApp";  
4  
5     public static void main(String[] args) {  
6         System.out.println("Nilai maksimum: " + MyApp1.MAX_VALUE);  
7         System.out.println("Nama aplikasi: " + MyApp1.APP_NAME);  
8     }  
9 }
```

Output - MyApp (run) ×

```
run:  
Nilai maksimum: 100  
Nama aplikasi: MyApp  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Method

Method adalah aksi atau tindakan yang dapat dilakukan oleh objek dari suatu kelas. Method beroperasi pada objek atau kelas dan dapat digunakan untuk melakukan tindakan tertentu seperti mengembalikan nilai, atau mengubah keadaan objek. Method berfungsi mengorganisir dan mengelompokkan kode, sehingga tindakan atau fungsi tertentu dapat dijalankan dengan cara yang terstruktur.

Jenis metode dalam OOP Java:

A. Instance Methods

Instance Methods adalah metode dalam OOP yang terkait langsung dengan objek spesifik yang diciptakan dari suatu kelas. Instance method berjalan pada atribut dan perilaku objek tertentu yang diwakili oleh instansiasi objek yang telah dibuat dari kelas tersebut.

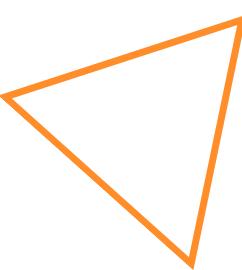
```
1 public class Mahasiswa {  
2     String nama;  
3     int umur;  
4  
5     // Metode instance untuk menampilkan informasi mahasiswa  
6     public void tampilanInfo() {  
7         System.out.println("Nama: " + nama);  
8         System.out.println("Umur: " + umur + " tahun");  
9     }  
10  
11    // Metode instance untuk mengubah umur mahasiswa  
12    public void ubahUmur(int newUmur) {  
13        umur = newUmur;  
14    }  
15  
16}  
17 }
```

```
1 public class main {  
2     public static void main(String[] args) {  
3         Mahasiswa mhs1 = new Mahasiswa();  
4         mhs1.nama = "Firman";  
5         mhs1.umur = 23;  
6         mhs1.tampilanInfo(); // Memanggil metode instance  
7         mhs1.ubahUmur(21); // Memanggil metode instance untuk mengubah umur  
8         mhs1.tampilanInfo(); // Memanggil metode instance lagi  
9     }  
10 }
```

Hasil :



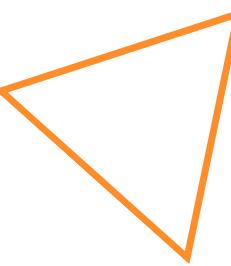
```
Output - MyApp (run) × Git - InventorySparepartMotor - master
▶ run:
▶ Nama: Firman
▶ Umur: 23 tahun
▶ Nama: Firman
▶ Umur: 21 tahun
▶ BUILD SUCCESSFUL (total time: 0 seconds)
```



Jenis metode dalam OOP Java:

B. Static Methods

Static methods adalah method yang terkait dengan kelas itu sendiri, bukan dengan objek yang dibuat dari kelas. method static diakses melalui nama kelas tanpa perlu membuat objek dari kelas tersebut. Karena method ini tidak beroperasi pada objek spesifik, mereka tidak dapat mengakses atribut non-static kelas atau variabel anggota non-static.



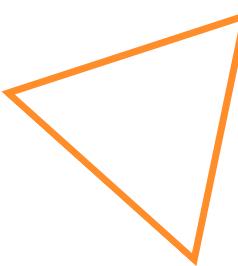
```
● ● ●

1 public class staticDemo {
2     static int x;
3     static int y;
4     static int hasil;
5
6     static int jumlah(){
7         hasil = x * y ;
8         return hasil;
9     }
10    public static void main(String[] args) {
11        staticDemo.x = 4;
12        staticDemo.y = 5;
13        System.out.println("Hasil Penjualan : "+staticDemo.jumlah());
14    }
15 }
```

Hasil :



```
Output - MyApp (run) ×  
run:  
Hasil Penjualan : 20  
BUILD SUCCESSFUL (total time: 0 seconds)
```



Jenis metode dalam OOP Java:

C. Final Methods

Pada metode final tidak dapat di-override oleh kelas turunan atau "child class". Final methods digunakan ketika ingin mencegah kelas turunan untuk mengubah perilaku khusus dari metode yang ada dalam kelas dasar. Method final digunakan untuk menjaga konsistensi perilaku metode.

```
public class Animal {  
    public final void suara() {  
        System.out.println("Hewan mengeluarkan suara.");  
    }  
  
    public class kucing extends Animal {  
        //Metode ini tidak dapat diubah karena final method di kelas dasar  
        //    public void suara() {  
        //        System.out.println("Kucing mengeluarkan suara: Meow!");  
        //    }  
    }  
}
```

```
public class exampleFinal {  
    public static void main(String[] args) {  
        kucing kcng = new kucing();  
        //karena method suara tidak di turunkan maka method kcng tersebut tidak bisa digunakan  
        //kcng.suara();  
    }  
}
```

Berdasarkan return type method :

A. Method Void (Procedure)

Merupakan method yang tidak memiliki nilai balik.

Syntak penulisan :

```
Void cetakHalo() //tipe_method nama_method
{
    System.out.println("HELLO WORLD!"); //badan method
}
```

The screenshot shows a Java code editor with a dark theme. At the top, there are three colored circular icons: red, yellow, and green. The code itself is as follows:

```
1 public class majalah {
2     //mendefinisikan variable
3     String judul = "Habis Gelap Terbitlah Terang";
4     String penulis = "Bagindo Dahlan Abdullah, dkk";
5     String penerbit = "Balai Pustaka";
6     int tahunProduksi = 2005 ;
7
8     //method menggunakan void
9     public void cetakMajalah() {
10        System.out.print("Judul :" + judul + "\n Penulis :" + penulis + "\n Penerbit :" + penerbit);
11        System.out.print("\n Tahun :" + tahunProduksi + "\n");
12    }
13    public static void main(String[] args) {
14        //membuat object
15        majalah mjl = new majalah();
16        mjl.cetakMajalah();
17    }
18 }
```

Hasil :

```
Output - MyApp (run) ×
run:
Judul :Habis Gelap Terbitlah Terang
Penulis :Bagindo Dahlan Abdullah, Zainudin Rasad, Sutan Muhammad Zain, dan Djamaloedin Rasad
Penerbit :Balai Pustaka
Tahun :2005
BUILD SUCCESSFUL (total time: 0 seconds)
```

Berdasarkan return type method :

B. Method Non-Void (Fungsi)

Merupakan method yang mempunyai nilai balik. Nilai yang dikembangkan sebagai hasil fungsi harus bertipe sama dengan tipe fungsi

```
int jumlahRoda(int x) //tipe_method nama_method(parameter)
{
    return x; //nilai kembalian
}
```

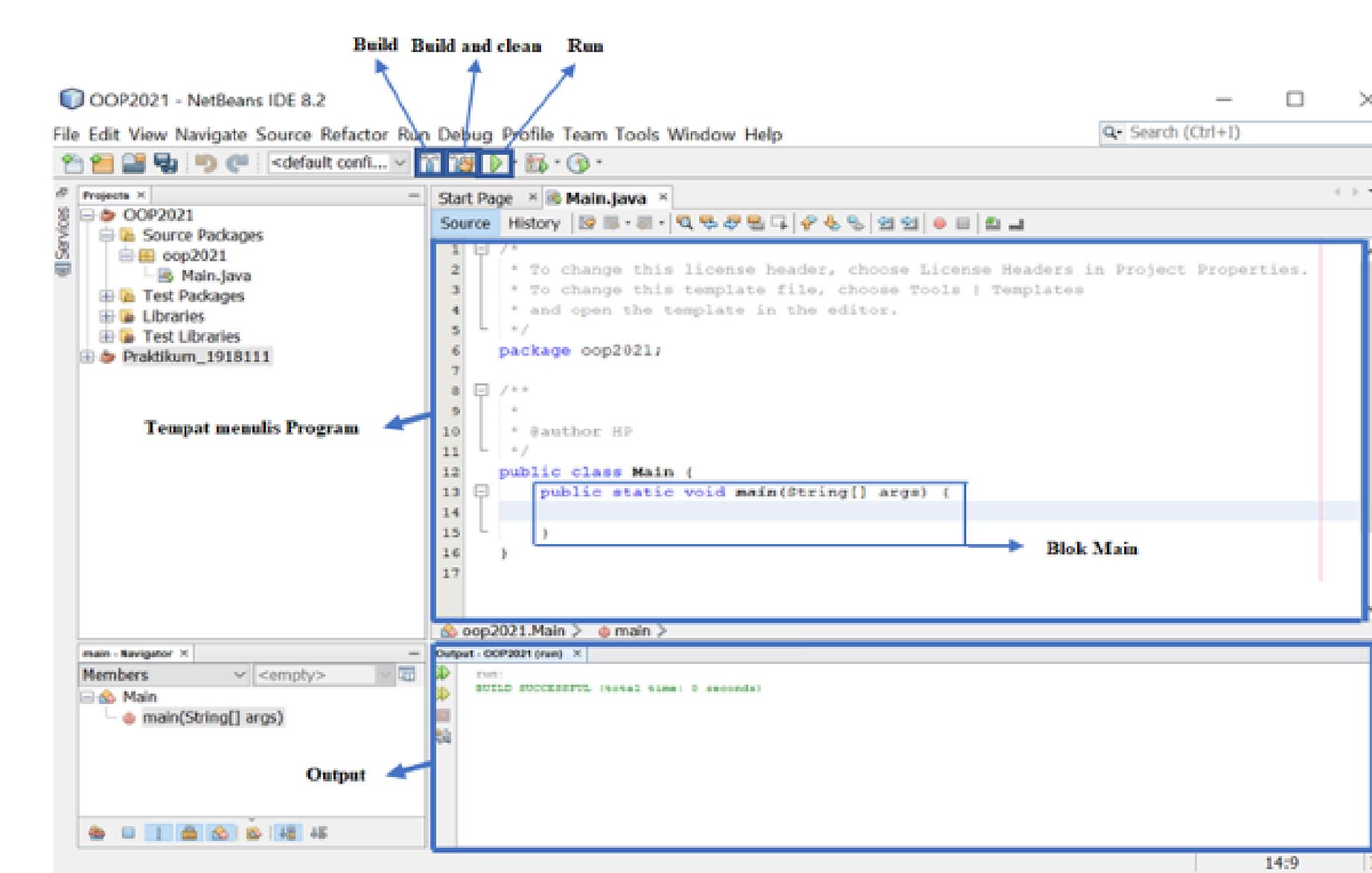
```
1  public class cal {
2      public int tambah(int a, int b) {
3          return a + b;
4      }
5      public int kurang(int a, int b) {
6          return a - b;
7      }
8      public int kali(int a, int b) {
9          return a * b;
10     }
11     public int bagi(int a, int b) {
12         return a / b;
13     }
14     public int modulus(int a, int b) {
15         return a % b;
16     }
17     public static void main(String[] args) {
18         cal method = new cal();
19         System.out.println("Penambahan: " + method.tambah(30, 15));
20         System.out.println("Pengurangan: " + method.kurang(30, 15));
21         System.out.println("Perkalian: " + method.kali(30, 15));
22         System.out.println("Pembagian: " + method.bagi(30, 15));
23         System.out.println("Modulus: " + method.modulus(30, 15));
24     }
25 }
```

Hasil :

```
Output - MyApp (run) ×
run:
Penambahan: 45
Pengurangan: 15
Perkalian: 450
Pembagian: 2
Modulus: 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

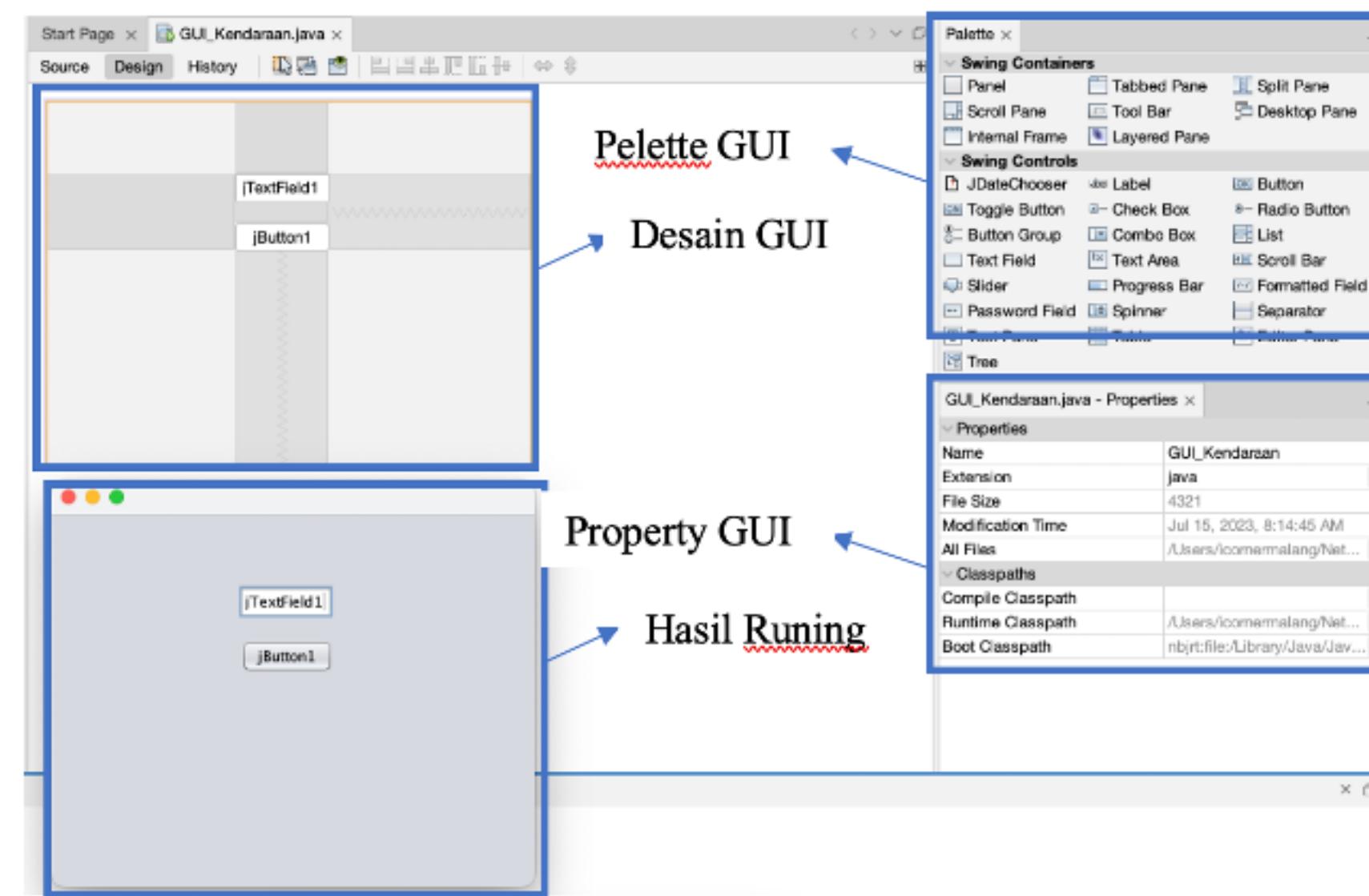
C. Compile dan Eksekusi Program menggunakan Netbeans

NetBeans adalah sebuah Integrated Development Environment (IDE) yang populer untuk pengembangan perangkat lunak, NetBeans menyediakan lingkungan yang lengkap dan intuitif untuk mengembangkan aplikasi Java, termasuk aplikasi desktop, web, dan mobile.



D .Pengenalan GUI pada neatbeans

GUI adalah antarmuka pengguna grafis yang memungkinkan pengguna berinteraksi dengan perangkat lunak atau sistem komputer menggunakan elemen visual seperti tombol, jendela, ikon, dan kontrol lainnya. Tujuannya adalah menambahkan beberapa komponen yang tidak bisa dibuat dalam basis text.



"Open NetBeans, and let's write code Java"