
# Class & Object

**Silver** - Chapter 1 - Topic 4

Selamat datang di Chapter 1 Topic 4 online course Android Developer dari Binar Academy!

## Haaii Binarian 👋

**Masih di Chapter 1 niih~**

Di Topik sebelumnya, kita sudah belajar tentang **Control Flow** dan **Operator**

Pada topik ini kita bakal belajar tentang **CLASS** dan **OBJECT** pada bahasa pemrograman Kotlin.

Nggak cuma itu~ kita semua akan coba praktekin formulanya bareng-bareng. Gimana guys, udah siaapp? 😃

## Detailnya, kita bakal bahas hal-hal berikut ini :

- Mengenal **Class** dan penggunaannya dalam Kotlin
- Mengenal **Object** dan cara membuatnya dalam Kotlin
- Mengenal **Constructor** dan penggunaannya dalam Kotlin

Oke, sekarang kita akan berkenalan dulu dengan **Class.**

Apa sih **Class** itu?

Yuk kita cek di slide selanjutnya~

Untuk bahan bacaan tambahan, kamu bisa cek disini~


Kalo di Kotlin, **class** ini bertindak sebagai *blueprint* atau cetak biru alias kerangka kerja yang fungsinya untuk membuat sebuah **property** dan **method.**

Dengan kata lain, semua yang ada di Kotlin bakal selalu dikaitin sama class dan object. Nah nggak cuma itu aja, seterusnya Kotlin juga bakal banyak berkaitan sama  yang namanya **property** dan **method.**

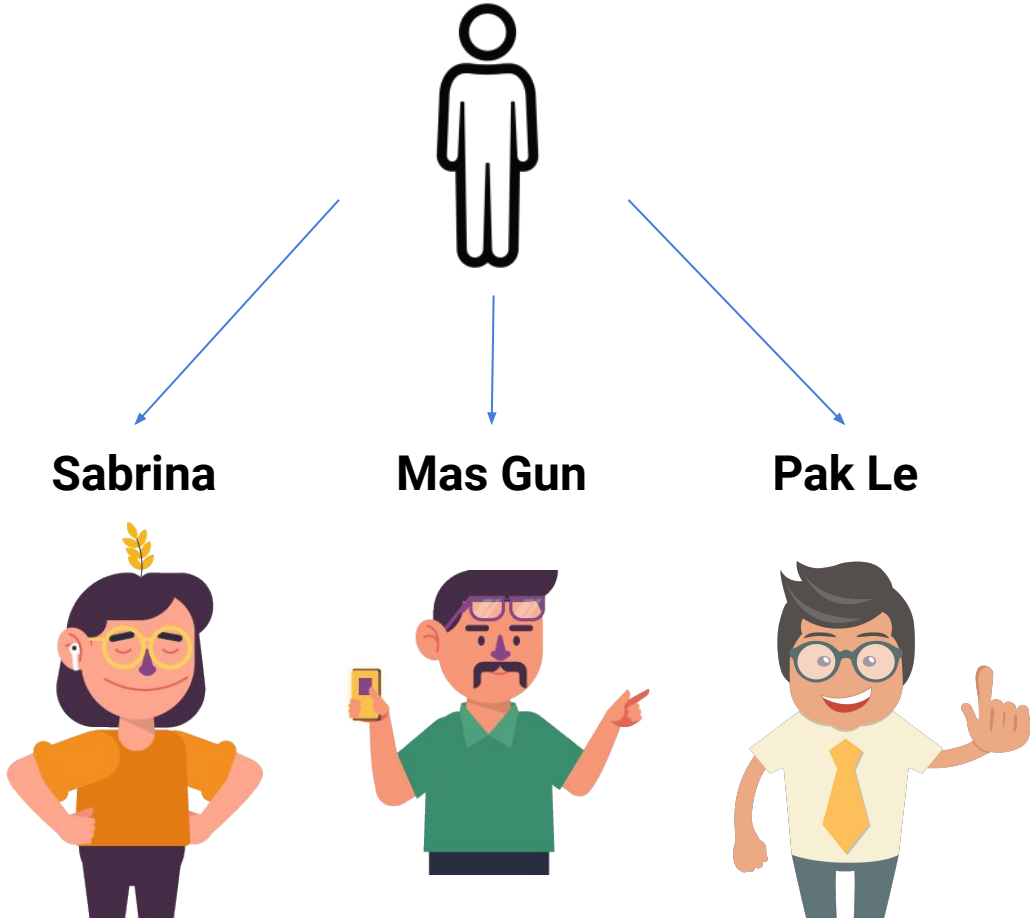
## Supaya lebih familiar kita coba pakai analogi sakti~

Coba kamu lihat gambar disamping!

Manusia bisa kita ibaratkan sebagai sebuah **CLASS.**

Manusia itu ada banyak karakter yang kita kenal. Misalnya, bestie kita ada **Sabrina, Mas Gun, Pak Le**.

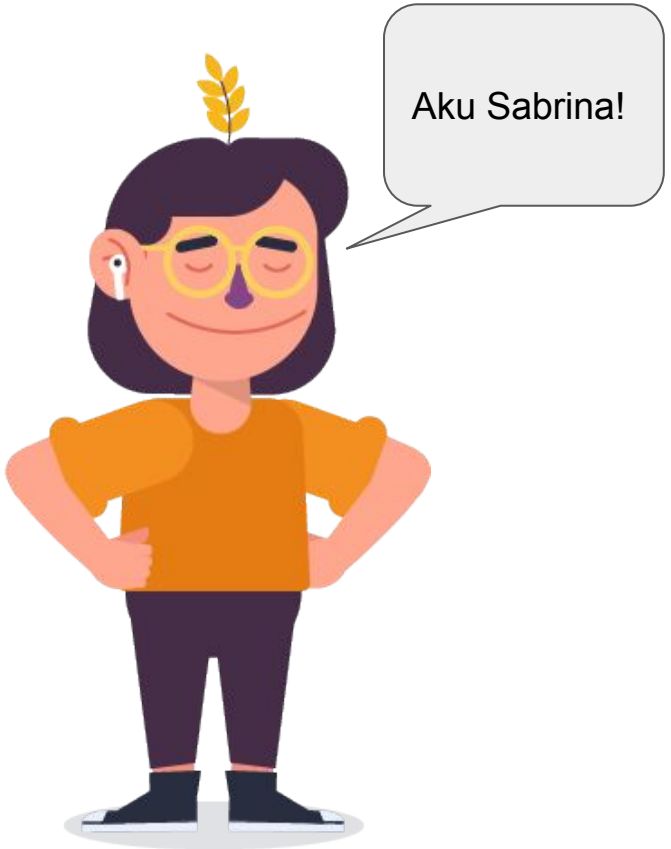
Kalau Sabrina, Mas Gun, dan Pak Le adalah bagian dari Manusia. Maka Sabrina, Mas Gun, dan Pak Le adalah **OBJECT-nya**dari **CLASS Manusia**



Cap cip cup kembang kuncup, kita pilih Sabrina buat nemenin kita belajar lebih spesifik.

Nama dia Sabrina, perempuan dengan tinggi badan 160 cm dan berat badan 58 kg.

Kalo di Kotlin, nama, jenis kelamin, tinggi badan, dan berat badan Sabrina ini yang kita sebut sebagai **PROPERTY.**



Dalam aktivitas kesehariannya, Sabrina ini demen banget jogging dan berenang.

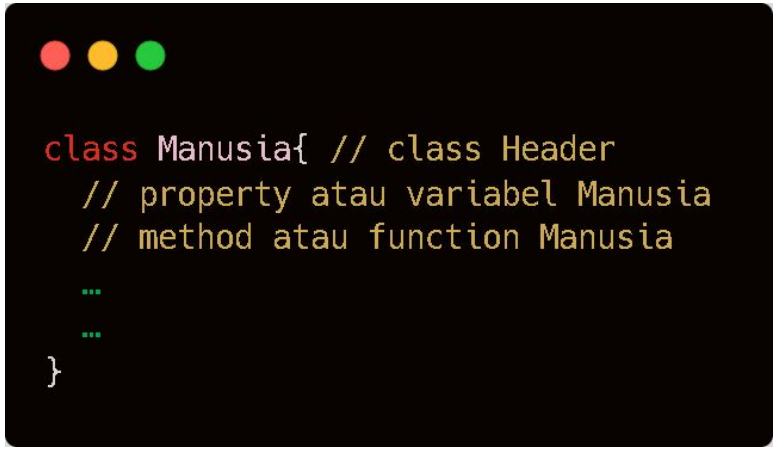
Nah, berbagai aktivitas yang dilakuin sama Sabrina ini kalo di Kotlin disebut sebagai **Method/Function**.

Setelah kita mengenal Class dan Object, kita akan mulai pelajarin satu per satu dengan lebih detail.

Pertama, kita cari tahu gimana caranya bikin **CLASS!**

## Buat bikin class, tinggal pakai *keyword* class

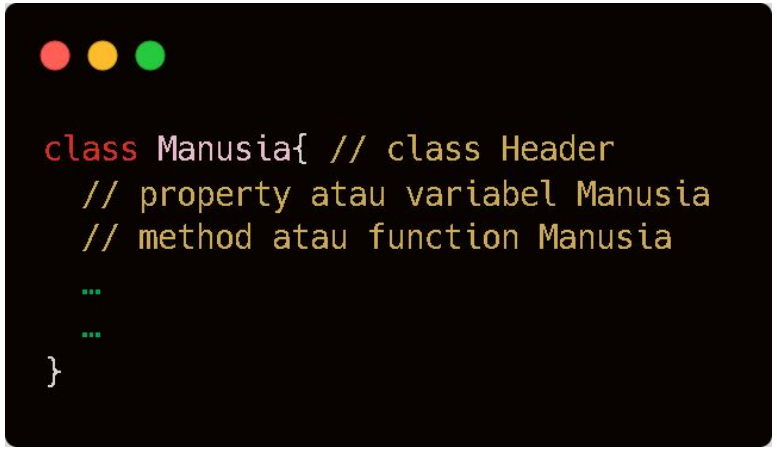
Pas kita bikin class dengan Kotlin, tipe default class-nya adalah **public** (class tersebut bisa dipanggil oleh class lain) Beda sama Java yang harus tentuin tipe class dulu baru bisa manggil clas lain.



## Bikin Class ada aturan penulisannya looh 😁

Pada bahasa pemrograman Kotlin ada sebuah **aturan penulisan** yang dianjurkan, yaitu setiap kali mau bikin **class** baru, kamu harus mengawali dengan huruf besar.

Coba intip contoh disamping, setelah diawali huruf besar harus dilanjutkan dengan huruf kecil. Selain itu, **Nama file** yang dipake buat nyimpen kode Kotlin juga harus sama kaya **nama Class** yang udah kita tulis ya!



Tadi kan cara bikin Class udah, sekarang kita coba masukin property untuk gabung di Class kita.

Gimana sih caranya ngasih **PROPERTY** ke Class?

## Buat ngasih Property ke Class, caranya begini ....

Mudah gan! Cara deklarasinya bisa cek kode disamping :

- **nama, kelamin,**
- **tinggiBadan,**
- **beratBadan**, dan
- **eat()**

Keempatnya dalah **Property** dari **class Manusia.**

```
class Manusia {
    var nama : String = "Sabrina"
    var kelamin : String = "Wanita"
    var tinggiBadan: Int = 170
    var beratBadan: Int = 60
    fun eat() = "Butuh Makan"
}
```

Setelah tau caranya untuk masukin **property** ke **Class,** sekarang kita tinggal masuk deh ke class yang udah dibuat.

Gimana sih **cara akses Class yang udah kita buat?**

## Buat akses Class, caranya begini ...

Untuk memanggil **class Manusia**, kita hanya perlu membuat variabel di **function main()** dan memasukkan **class Manusia** di dalamnya.

Setelah itu kita dapat memanggil semua property ataupun function yang terdapat **class Manusia**.

Nah pemanfaatan dari **class** tersebut nantinya yang disebut sebagai **object**.

```
class Manusia {
    var nama : String = "Sabrina"
    var kelamin : String = "Wanita"
    fun eat() = "Butuh Makan"
}

fun main() {
    val manusia = Manusia()
    println("Nama : ${manusia.nama}")
    println("Kelamin : ${manusia.kelamin}")
    println("Makan : ${manusia.eat()}")
}
```

## Apa itu OBJECT?

Object adalah representasi dari class itu sendiri.

Supaya lebih memahami object sampai ke akarnya, bisa intip dulu link disini ya bestie~

# OBJECT

Object merupakan sebuah *instance* dari suatu class. Kalo tadi kita udah bahas soal class, nah, object ini **fungsinya buat ngambil isi  dari dalam class.**

Salah satu fungsi dari object itu buat akses berbagai **function** atau **method** yang ada di suatu class

Sama kayak Class tadi, pertama-tama kita akan belajar cara bikin Object-nya dulu.

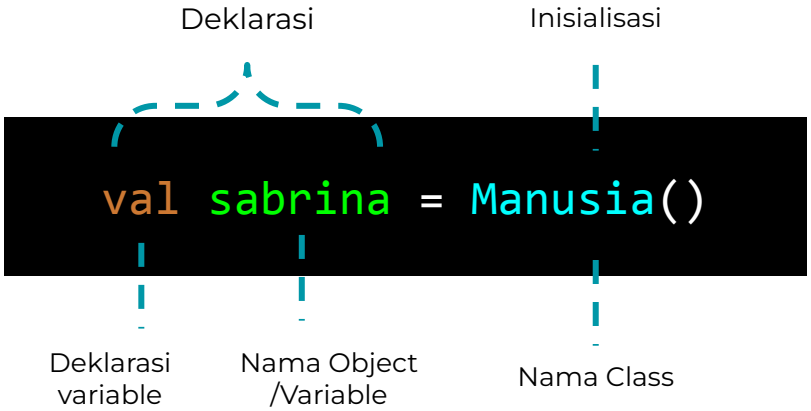
Gimana sih cara bikin OBJECT?

## Cara Bikin Object, Begini nih ...

Buat bikin object, kita perlu mendeklarasikan dulu tiap object yang dibuat dengan mengikutsertakan class bawaannya.

Ada dua tahap yang harus dilakukan buat bikin sebuah object. Tahap tersebut yakni :

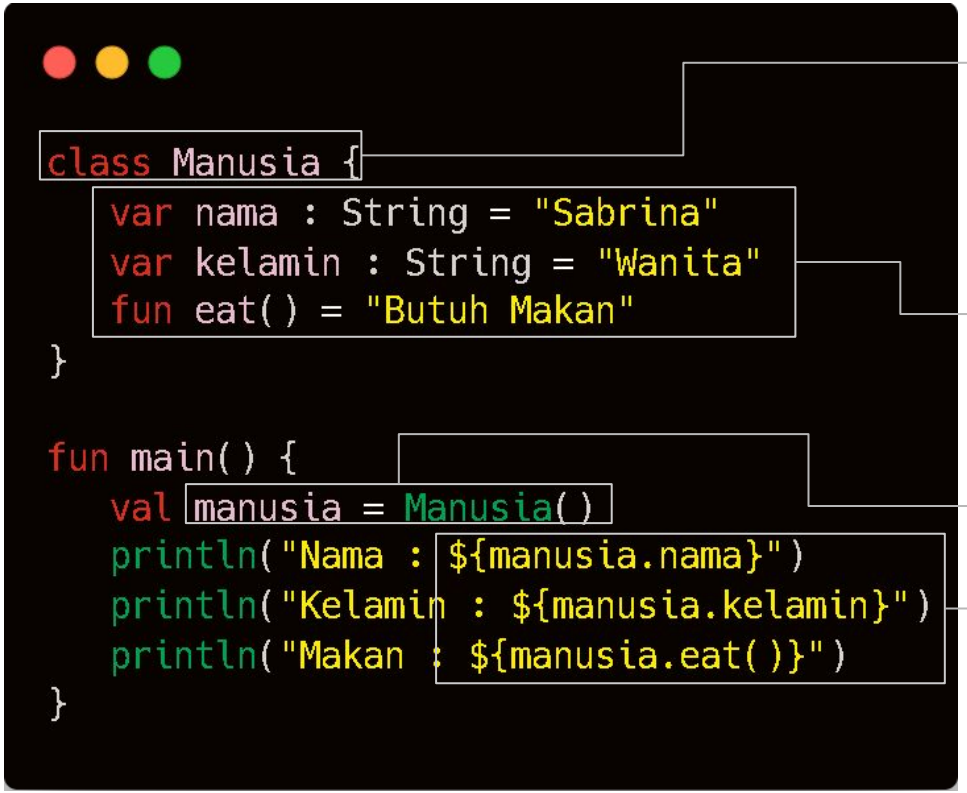
- **Declaration**
  Mendeklarasikan nama object yang akan dibuat.
- **Initialization**
  Memasukkan class yang akan digunakan sebagai Object




Constructor method :

- Method yang pertama kali dijalankan ketika sebuah object dibuat.
- **Nama Method** Constructor **HARUS** sama persis seperti **Class Name**
- Secara default Constructor seperti NamaKelas();

## Coba kita Overview dulu cara deklarasinya



```
class Manusia {
    var nama : String = "Sabrina"
    var kelamin : String = "Wanita"
    fun eat() = "Butuh Makan"
}

fun main() {
    val manusia = Manusia()
    println("Nama : ${manusia.nama}")
    println("Kelamin : ${manusia.kelamin}")
    println("Makan : ${manusia.eat()}")
}
```

*Class* bernama **Manusia**

Class **Manusia** memiliki property **nama** dengan :

- tipe data **String** dengan value **Sabrina**,
- **kelamin** dengan tipe data **String** dengan value **Wanita**,
- dan sebuah function dengan nama **eat()** dengan value “**Butuh Makan**”.

**manusia** adalah nama object yang kita buat dari class Manusia.

Untuk memanggil sebuah **method** di dalam kelas, kita dapat menggunakan “.” (titik).

## Catatan tentang Object

Membuat Object pada Kotlin juga bisa dengan membuat file object terpisah dan menambahkan beberapa function sesuai yang diinginkan seperti gambar disamping :

Dan untuk memanggilnya, kita hanya perlu memanggil file object yang ingin kita panggil, dan masukkan function mana yang ingin kita gunakan seperti gambar dibawah :

```
object ContohObject {
    fun contoh1(): String {
        return "contoh 1"
    }

    fun contoh2(): String {
        return "contoh 2"
    }
}
```

```
fun main() {
    ContohObject.contoh1()
    ContohObject.contoh2()
}
```

Nice! Sampe di sini pengetahuan kamu tentang cara membuat class dan object sudah meningkat, nih. Ciyee~

Mudah kan memahami konsep Class dan Object?

Sekarang kita akan pelajari constructor yang bisa digunakan sebagai persiapan awal pada dua hal tersebut.

Penjelasan lebih lanjut bisa cek disini banget guys!

## Apa itu Constructor?

Setelah tahu cara membuat **class** dan juga **object**, kita juga perlu mengenal apa yang namanya **constructor**.

Ketika membuat suatu **object**, semua properti pada **class** haruslah memiliki nilai. Kita dapat membuat sebuah nilai atau insialisasi class dengan **constructor**.

**Constructor** merupakan fungsi spesial yang digunakan untuk menginisialisasi **properti** yang terdapat pada **class** tersebut.

Kotlin membagi Constructor menjadi 2 tipe:

- Primary Constructor
- Secondary Constructor

Primary constructor dapat di deklarasikan di dalam **header class**. Hal tersebut memudahkan kita untuk request sebuah object yang nantinya akan memanggil class tersebut.

Contohnya? Yuk kita simak slide berikutnya 😆

## Cara deklarasi Primary Constructor

Ketika membuat sebuah **class**, kita diwajibkan mengisi setiap **properti** yang ada. Lantas bagaimana cara membuat **class** yang dapat diisi secara dinamis? Di situ lah peran **constructor** dibutuhkan. Kita dapat membuat sebuah **class** seperti berikut :

```
class Manusia(
    val nama: String,
    val kelamin: String,
    val tinggiBadan: Int,
    val beratBadan: Int
) {
  // blok body class
}
```

constructor

```
fun main() {

    val mahasiswa = Manusia(
        nama = "Sabrina",
        kelamin = "Wanita",
        tinggiBadan = 170,
        beratBadan = 60
    )

    println("Nama : ${mahasiswa.nama}")
    println("Kelamin : ${mahasiswa.kelamin}")
    println("Kelamin : ${mahasiswa.tinggiBadan}")
    println("Kelamin : ${mahasiswa.beratBadan}")
}
```

Nah untuk memanggilnya kita dapat deklarasi kode nya seperti disamping

## Apa itu Secondary Constructor?

Kotlin juga memungkinkan kita untuk membuat satu atau beberapa **secondary constructor** pada sebuah class. Secondary constructor ini dibuat menggunakan kata kunci **constructor**.

Hal tersebut diperlukan setiap kali kita ingin membuat lebih dari satu **constructor** di Kotlin atau kapan pun kita ingin memasukkan lebih banyak logika dalam **primary constructor** namun tidak dapat dilakukan karena **primary constructor** dapat dipanggil oleh beberapa class lain.

Kalau kode disamping dijalankan, hasilnya akan seperti dibawah ini :

```
Initializer Block
Nama = Sabrina

Tinggi Badan = 160
```

Dalam **secondary constructor** tidak diijinkan untuk menggunakan **val** atau **var** dengan parameter **secondary constructor.**

Blok ini contoh deklarasi **secondary constructor**-nya

```
class Manusia {
    // Member Variables
    var nama: String
    var tinggiBadan: Int

    // Initializer Block
    init {
        println("Initializer Block")
    }

    // Secondary Constructor
    constructor (_nama: String, _tinggi: Int) {
        this.nama = _nama
        this.tinggiBadan = tinggi
        println("Nama = $nama")
        println("Tinggi Badan = $tinggiBadan")
    }
}

fun main() {
    val zara = Manusia("Sabrina", 20)
}
```

## Saatnya kita

# Quiz!

**1. Berikut ini adalah berbagai fakta tentang Class dalam Pemrograman Kotlin, kecuali ....**

A. Sebuah blueprint atau cetak biru yang fungsinya untuk menampung property dan function

B. Secara default, Class jika dideklarasikan akan bersifat public

C. Sebuah class yang menampung berbagai child class

BINAR

**1. Berikut ini adalah berbagai fakta tentang Class dalam Pemrograman Kotlin, kecuali ....**

A. Sebuah blueprint atau cetak biru yang fungsinya untuk menampung property dan function

B. Secara default, Class jika dideklarasikan akan bersifat public

C. **Sebuah class yang menampung berbagai child class**

**Class** pada Kotlin bertindak sebagai blueprint atau kerangka kerja yang fungsinya untuk menampung **property** dan **function**.
**Class** juga bisa memiliki beberapa **property** didalamnya.

BINAR

**2. Kira-kira syntax seperti apa yang cocok kita gunakan untuk mengisi kekosongan di atas, agar hasil yang ditampilkan sesuai dengan soal?**

```
object Mahasiswa {
    fun nama(): String {
        return "Binarian"
    }

    fun usia(): Int {
        return 25
    }
}

fun main() {
    val result = //… isi di sini
    println(result) // Result "Binarian Jomblo"
}
```

A. "${Mahasiswa.nama()} ${Mahasiswa.usia()}"

B. "Mahasiswa.nama() Mahasiswa.usia()"

C. Mahasiswa.nama() Mahasiswa.usia()

BINAR

**2. Kira-kira syntax seperti apa yang cocok kita gunakan untuk mengisi kekosongan di atas, agar hasil yang ditampilkan sesuai dengan soal?**

```
object Mahasiswa {
    fun nama(): String {
        return "Binarian"
    }

    fun usia(): Int {
        return 25
    }
}

fun main() {
    val result = //… isi di sini
    println(result) // Result "Binarian Jomblo"
}
```

A. **"${Mahasiswa.nama()} ${Mahasiswa.usia()}"**

B. **"Mahasiswa.nama() Mahasiswa.usia()"**

C. **Mahasiswa.nama() Mahasiswa.usia()**

Agar dapat mengeluarkan output `"Binarian Jomblo"`
Jawaban A adalah yang benar, karena kita sudah memanggil object **Mahasiswa** dan memanggil function **nama** dan function **status** yang ada di dalam object tersebut.

BINAR

**3. Dari pernyataan-pernyataan dibawah ini mengenai Constructor, manakah yang benar?**

**A.** Merupakan fungsi spesial yang digunakan untuk menginisialisasi property yang terdapat pada class tersebut.

**B.** Dalam Kotlin, Constructor dibagi menjadi dua tipe: Primary Constructor dan Sub-Constructor

**C.** Untuk mendeklarasikan Primary Constructor, gunakan tanda titik “.” setelah property

BINAR

**3. Dari pernyataan-pernyataan dibawah ini mengenai Constructor, manakah yang benar?**

A. Merupakan fungsi spesial yang digunakan untuk menginisialisasi property yang terdapat pada class tersebut.

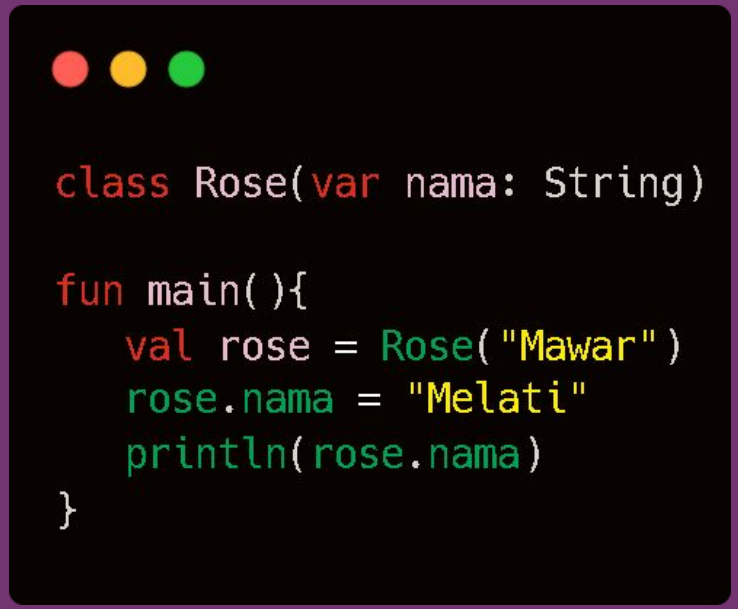
B. **Dalam Kotlin, Constructor dibagi menjadi dua tipe: Primary Constructor dan Sub-Constructor**

C. Untuk mendeklarasikan Primary Constructor, gunakan tanda titik “.” setelah property

Seperti yang sudah kita pelajari di slide sebelumnya, bahwa **constructor** merupakan fungsi spesial yang digunakan untuk menginisialisasi **property** yang terdapat pada class tersebut.

BINAR

**4. Perhatikan syntax berikut, Apakah output yang akan dihasilkan?**



```
class Rose(var nama: String)

fun main(){
    val rose = Rose("Mawar")
    rose.nama = "Melati"
    println(rose.nama)
}
```

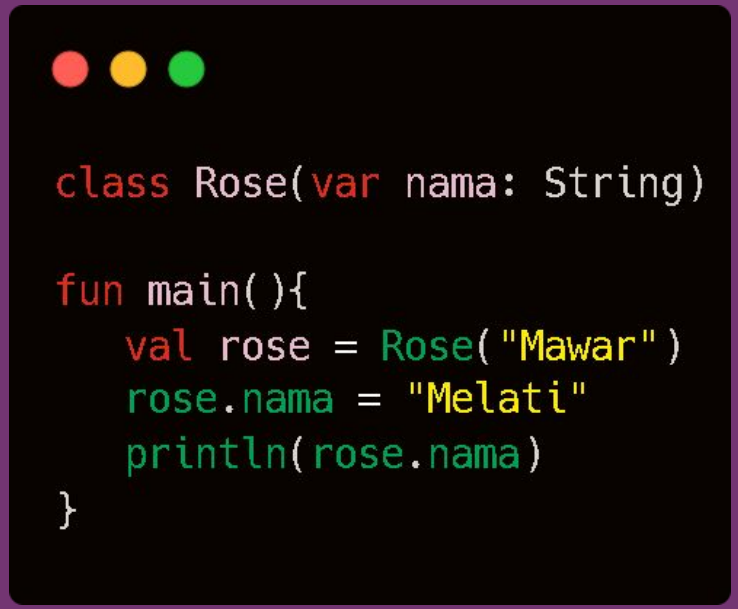
A. Mawar

B. Melati

C. Error

BINAR

## 4. Perhatikan syntax berikut, Apakah output yang akan dihasilkan?



```
class Rose(var nama: String)

fun main(){
    val rose = Rose("Mawar")
    rose.nama = "Melati"
    println(rose.nama)
}
```

A. Mawar

**B. Melati**

C. Error

Tepat sekali!!!

Jawaban yang benar adalah Melati. Itu dikarenakan kita mengisi ulang **variable** nama menjadi **Melati** di akhir.

BINAR

## 5. Apa yang akan terjadi ketika kita menjalankan syntax di bawah ini?

```
class Test {
    var i : Int
}

fun main(){
    val test = Test()
    test.i = 19
    println(test.i)
}
```

A. 19

B. Runtime Error

C. Property must be initialized or be abstract

BINAR

## 5. Apa yang akan terjadi ketika kita menjalankan syntax di bawah ini?

```
class Test {
    var i : Int
}

fun main(){
    val test = Test()
    test.i = 19
    println(test.i)
}
```

A. 19

B. Runtime Error

C. **Property must be initialized or be abstract**

Ketika kita membuat sebuah **constructor**, kita harus menginisialisasi **property** tersebut dengan benar. Barulah kita dapat menggunakannya.

## Referensi dan bacaan lebih lanjut~

1. Definisi Class
2. Definisi Object
3. Definisi Constructor

Nah, selesai sudah pembahasan kita di Chapter 1 Topic 4 ini.

Selanjutnya, kita bakal bahas **Function dan Parameter** yang banyak digunakan dalam penyusunan aplikasi Android.

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~

# Terima Kasih!

## Next Topic

loading...

BINAR