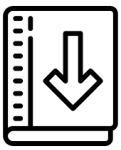
MODUL 2
DASAR PEMROGRAMAN KOTLIN, FUNGSI DAN KELAS

CAPAIAN PEMBELAJARAN

1. Mahasiswa mampu mengimplementasikan dasar-dasar pemrograman dengan Kotlin
2. Mahasiswa mampu mengimplementasikan fungsi dan dipanggil dalam program

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. https://play.kotlinlang.org/ untuk menjalankan program Kotlin.



DASAR TEORI

Fungsi adalah blok pernyataan terkait yang bersama-sama melakukan tugas tertentu. Sebagai contoh katakanlah kita harus menulis tiga baris kode untuk menghasilkan rata-rata dua angka, jika kita membuat fungsi untuk menghasilkan rata-rata maka kita tidak perlu menulis tiga baris itu lagi dan lagi, kita bisa memanggil fungsi yang kita buat.

Ada dua jenis fungsi di Kotlin:

1. Fungsi pustaka standar
2. Fungsi yang didefinisikan pengguna

Fungsi Pustaka Standar

Fungsi yang sudah ada di pustaka standar Kotlin disebut fungsi pustaka standar atau fungsi bawaan atau fungsi yang telah ditentukan. Misalnya ketika kita perlu menggunakan fungsi Math.floor() kita tidak mendefinisikan fungsi karena sudah ada dan kita bisa langsung memanggilnya dalam kode kita.

Fungsi Yang Didefinisikan Pengguna

Fungsi yang kita definisikan dalam program sebelum kita memanggilnya dikenal sebagai fungsi yang didefinisikan pengguna. Sebagai contoh, katakanlah kita ingin fungsi cek genap atau ganjil dalam program kita maka kita dapat membuat fungsi untuk tugas ini dan kemudian memanggil fungsi dimana kita perlukan untuk melakukan cek genap atau ganjil.

Kita membuat fungsi menggunakan kata kunci fun. Mari kita membuat fungsi yang mencetak "Halo".

```
//Created the function
fun sayHello(){
    println("Halo")
}
fun main(args : Array<String>){

    //Calling the function
    sayHello()
}
```

Fungsi yang didefinisikan pengguna dengan argumen dan tipe kembali, dengan sintaks:

```
fun function_name(param1: data_type, param2: data_type, ...): return_type
```

Contoh:

```
fun jarak(x:Int, y:Int): Double
{
    var hasil = Math.sqrt(x*x*1.0 + y*y)
    return hasil
```

```
}

fun main(args : Array<String>){
    println("Jarak dari (0,0): ${jarak(3, 4)}")
}
```

Fungsi ini menerima argumen/parameter dengan jumlah yang variabel (bisa berapa saja), kita gunakan kata kunci **vararg**. Perhatikan contoh berikut.

```
fun jumlah(vararg bil2: Int): Int
{
    var jml = 0
    bil2.forEach {bil ->
        jml +=  bil
    }
    return jml
}
```

Contoh pemanggilan fungsi ini bisa seperti berikut.

```
println("Jumlah bilangan: ${jumlah(10, 20, 30, 40)}")
```

Fungsi Inline

Fungsi Inline (disebut juga fungsi lambda) dapat didefinisikan di dalam fungsi main (). Mari kita ambil contoh fungsi inline. Dalam contoh berikut ini kita telah mendefinisikan fungsi inline jumlah() yang menerima dua argumen integer bil1 dan bil2 dan tipe hasil adalah integer.

```
fun main(args : Array<String>){
    val jumlah = {bil1: Int, bil2: Int -> bil1 + bil2}
    println("6 + 4 = ${jumlah(5,4)}")
}
```

Higher-Order Function

Fungsi orde tinggi (higher-order function) dapat memiliki fungsi lain sebagai parameter atau mengembalikan fungsi atau dapat melakukan keduanya. Sampai sekarang kita telah melihat bagaimana kita meneruskan bilangan bulat, string dll sebagai parameter untuk suatu fungsi tetapi dalam modul ini, kita akan belajar

bagaimana kita melewatkan suatu fungsi ke fungsi lain. Kita juga akan melihat bagaimana suatu fungsi mengembalikan fungsi lainnya.

Dalam contoh berikut ini, kami melewatkan fungsi demo() ke fungsi func lainnya (). Untuk meneruskan fungsi sebagai parameter ke fungsi lain, kita gunakan operator :: di depan fungsi seperti yang ditunjukkan pada contoh berikut.

```
fun main(args: Array<String>) {
    func("BeginnersBook", ::demo)
}

fun func(str: String, myfunc: (String) -> Unit) {
    print("Welcome to Kotlin tutorial at ")
    myfunc(str)
}

fun demo(str: String) {
    println(str)
}
```

Dalam contoh berikut ini fungsi `func` mengembalikan fungsi lain. Untuk memahami kode ini, mari kita lihat fungsi `func` terlebih dahulu, ia menerima parameter integer `num` dan di area kembali kita telah mendefinisikan fungsi:

```
(Int) -> Int = {num2 -> num2 + num}
```

Jadi ini adalah fungsi lainnya yang juga menerima parameter integer dan mengembalikan jumlah parameter dan `num` ini.

Anda mungkin bertanya-tanya mengapa kita telah melewatkan nilai 20 sebagai parameter dalam sum, nah ini karena fungsi func mengembalikan fungsi sehingga sum adalah fungsi yang akan menerima parameter int. Ini adalah fungsi yang sama yang telah kami definisikan di area pengembalian fungsi func.

```
fun main(args: Array<String>) {

    val sum = func(10)
    println("10 + 20: ${sum(20)}")

}

fun func(num: Int): (Int) -> Int = {num2 -> num2 + num}
```

Kotlin Class dan Objects

Kotlin adalah bahasa pemrograman berorientasi objek seperti Java. Pemrograman berorientasi objek (OOP) memungkinkan kita untuk memecahkan masalah yang kompleks dengan menggunakan objek.

Kelas adalah blok bangunan utama dari setiap bahasa pemrograman berorientasi objek. Semua objek adalah bagian dari kelas dan berbagi properti umum dan perilaku yang didefinisikan oleh kelas dalam bentuk data anggota dan fungsi anggota masing-masing.

Kelas didefinisikan menggunakan kata kunci `class` di Kotlin. Sebagai contoh:

```
class Contoh {

    // data member
    private var bilangan: Int = 5

    // member function
    fun hitungKuadrat(): Int {
        return bilangan * bilangan
    }
}
```

Kita belum menentukan pengubah akses (access modifier) apa pun pada kelas di atas. Pengubah akses membatasi akses. Secara default pengubah akses bersifat publik. Dalam contoh di atas kita belum menentukan pengubah akses apa pun sehingga secara default pengubah akses public berlaku untuk kelas `Contoh` di atas.

Access modifier:

- private – hanya dapat diakses didalam kelas.
- public – dapat diakses dimana saja
- protected – dapat diakses didalam kelas dan subkelasnya.
- internal – dapat diakses didalam modul.

Bagaimana menciptakan objek kelas

Objek kelas dapat diciptakan seperti berikut.

```
val c : Contoh = Contoh()
```

Untuk mengakses anggota dari kelas dapat dilakukan seperti berikut.

```
println("hasil : ${c.hitungKuadrat()}")
```

Konstruktor

Tujuan utama konstruktor adalah menginisialisasi properti kelas. Konstruktor dipanggil ketika kita membuat objek kelas.

Tipe Konstruktor:

- Konstruktor Utama - Menginisialisasi properti kelas
- Konstruktor Sekunder - Menginisialisasi properti kelas, kita dapat memiliki kode inisialisasi tambahan di dalam konstruktor sekunder.

Konstruktor utama adalah cara termudah untuk menginisialisasi kelas. Itu dinyatakan sebagai bagian dari header kelas. Dalam contoh berikut ini kita telah mendeklarasikan konstruktor (val nama: String, var umur: Int) sebagai bagian dari header kelas. Ini adalah konstruktor utama kita yang menginisialisasi properti nama dan umur (anggota data) dari kelas Mahasiswa.

```
fun main(args: Array<String>) {

    //creating the object of class Student
    val mhs = Mahasiswa("Susi Susanti", 23)

    println("Nama : ${mhs.nama}")
    println("Umur : ${mhs.umur}")
}

class Mahasiswa(val nama: String, var umur: Int) {
    //This is my class. For now I am leaving it empty
}
```

Kita juga dapat menentukan nilai default di konstruktor seperti contoh berikut.

```
class Mahasiswa(val nama: String = “Mahasiswa”, var umur: Int = 20) {
    //This is my class. For now I am leaving it empty
}
```

Dengan adanya nilai default, objek kelas dapat diciptakan tanpa menyertakan parameter.

```
    val mhs = Mahasiswa()
```

Kita dapat menambahkan kode initializer tambahan di dalam konstruktor suatu kelas. Blok initializer kita tulis di dalam konstruktor menggunakan init. Dalam blok ini kita dapat memiliki logika inisialisasi tambahan.

```
fun main(args: Array<String>) {
    val mhs = Mahasiswa("Susi Susanti", 23)
    val mhs2 = Mahasiswa("Haryanto", 22)
    val mhs3 = Mahasiswa()
}

class Mahasiswa(val nama: String = "Mahasiswa", var umur: Int = 99) {
    val namaMhs: String
    var umurMhs: Int
    init{
        if(nama == "Mahasiswa"){
           namaMhs = "Kosong"
           umurMhs = 0
        } else {
           namaMhs = nama.toUpperCase()
           umurMhs = umur
        }
        println("Nama : $namaMhs")
       println("Umur : $umurMhs")
    }
}
```

Kelas Data Kotlin

Di Kotlin, Anda bisa membuat kelas data untuk menyimpan data. Alasan mengapa Anda ingin menandai kelas sebagai data adalah untuk memberi tahu kompiler bahwa Anda membuat kelas ini untuk menyimpan data, kompiler kemudian membuat beberapa fungsi secara otomatis untuk kelas data Anda yang akan sangat membantu dalam mengelola data.

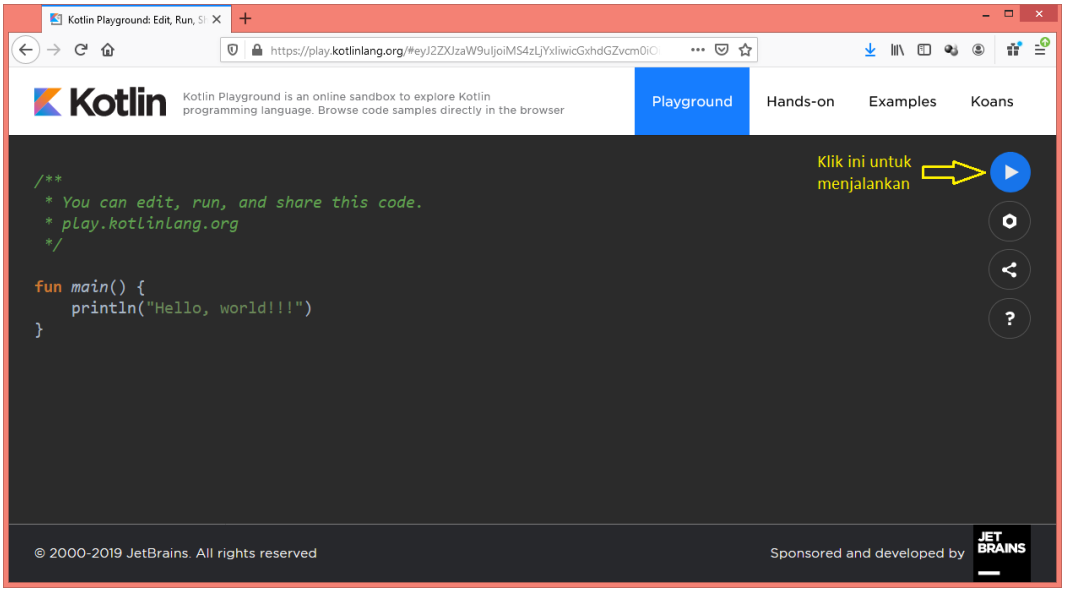
Karena kita telah mendeklarasikan kelas ini sebagai data, kompiler secara otomatis telah membangkitkan beberapa fungsi seperti copy(), toString(), equals () dll.

```
data class Mahasiswa(val nama: String, val umur: Int)
```

```
fun main(args: Array<String>) {
    val mhs = Mahasiswa("Susi Susanti", 23)
    val mhs2 = Mahasiswa("Agus Budianto", 25)
    println("Nama Mahasiswa: ${mhs.nama}")
    println("Umur Mahasisa:  ${mhs.umur}")
    println("Nama Mahasiswa: ${mhs2.nama}")
    println("Umur Mahasisa:  ${mhs2.umur}")
}
```

PRAKTIK

1. Anda akan menggunakan compiler Kotlin secara online. Lewat browser buka url: **play.kotlinlang.org** dengan tampilan seperti berikut:



2. Modifikasilah kode yang ditampilkan sehingga menjadi seperti berikut.

```
fun main(args : Array<String>){
    var num = 16
    println("Akar $num adalah: ${Math.sqrt(num.toDouble())}")
    val a = 12
    val b=15
    println("Nilai terbesar dar $a dan $b adalah: "+Math.max(a, b))
```

```
    println("4 pangkat 3 = ${Math.pow(4.0, 3.0)}")
}
```

3. Jalankan program yang ada dengan klik tombol lingkaran biru di sebelah kanan atas. Perhatikan hasil running program di bagian bawah jendela browser.

4. Buat program seperti berikut, dan perhatikan hasil running program.

```
//Created the function
fun jumlah(bilangan2: Array<Int>): Int
{
    var jml = 0
    for(bil in bilangan2){
        jml += bil
    }
    return jml
}

fun main(args : Array<String>){
    val arrBil = arrayOf(10, 20, 30, 50)
    println("Jumlah bilangan: ${jumlah(arrBil)}")
}
```

5. Modifikasilah program di atas sehingga argumen/parameter fungsi bersifat variabel, seperti berikut. Perhatikan hasil program.

```
fun jumlah(vararg bil2: Int): Int
{
    var jml = 0
    bil2.forEach {bil ->
        jml +=  bil
    }
    return jml
}

fun main(args : Array<String>){
    println("Jumlah bilangan: ${jumlah(10, 20, 30, 40)}")
}
```

6. Tambahkan jumlah parameter fungsi jumlah ini pada saat dipanggil, menjadi: jumlah(10, 20, 30, 40, 50, 60). Perhatikan hasil program.

7. Buat program seperti berikut yang menggunakan fungsi inline: jumlah, dan perhatikan hasil running program.

```
fun main(args : Array<String>){
    val jumlah = {bil1: Int, bil2: Int -> bil1 + bil2}
    println("6 + 4 = ${jumlah(5,4)}")
}
```

8. Modifikasilah program di atas yang digunakan untuk mengalikan 3 buah bilangan.
9. Jalankan program berikut dalam fungsi main(), perhatikan apa yang terjadi.

```
val upperCase1: (String) -> String = { str: String -> str.toUpperCase() }
val upperCase2: (String) -> String = { str -> str.toUpperCase() }
val upperCase3 = { str: String -> str.toUpperCase() }
val upperCase4: (String) -> String = { it.toUpperCase() }
val upperCase5: (String) -> String = String::toUpperCase

println(upperCase1("hello"))
println(upperCase2("hello"))
println(upperCase3("hello"))
println(upperCase4("hello"))
println(upperCase5("hello"))
```

10. Buat program seperti berikut, dan perhatikan hasil running program.

```
fun main(args: Array<String>) {
    func("BeginnersBook", ::demo)
}

fun func(str: String, myfunc: (String) -> Unit) {
    print("Welcome to Kotlin tutorial at ")
    myfunc(str)
}

fun demo(str: String) {
    println(str)
}
```

11. Buat program seperti berikut, dan perhatikan hasil running program.

```
fun main(args: Array<String>) {

    val sum = func(10)
    println("10 + 20: ${sum(20)}")

}

fun func(num: Int): (Int) -> Int = {num2 -> num2 + num}
```

12. Buat program seperti berikut, dan perhatikan hasil running program.

```
fun main(args: Array<String>) {

    //creating the object of class Student
    val mhs = Mahasiswa("Susi Susanti", 23)

    println("Nama : ${mhs.nama}")
    println("Umur : ${mhs.umur}")
}

class Mahasiswa(val nama: String, var umur: Int) {
    //This is my class. For now I am leaving it empty
}
```

13. Tambahkan objek Mahasiswa lagi dengan nama mhs2 dan berilah data nama dan umur, kemudian tampilkan.

14. Tambahkan nilai default pada konstruktor. Kemudian buatlah objek Mahasiswa (misal mhs3) tanpa data nama dan umur, dan tampilkan hasilnya.

15. Buat program seperti berikut, dan perhatikan hasil running program.

```
fun main(args: Array<String>) {
    val mhs = Mahasiswa("Susi Susanti", 23)
    val mhs2 = Mahasiswa("Haryanto", 22)
    val mhs3 = Mahasiswa()
}

class Mahasiswa(val nama: String = "Mahasiswa", var umur: Int = 99) {
    val namaMhs: String
    var umurMhs: Int
    init{
        if(nama == "Mahasiswa"){
           namaMhs = "Kosong"
           umurMhs = 0
        } else {
           namaMhs = nama.toUpperCase()
           umurMhs = umur
        }
        println("Nama : $namaMhs")
           println("Umur : $umurMhs")
    }
}
```

16. Buat program seperti berikut, dan perhatikan hasil running program.

```
data class Mahasiswa(val nama: String, val umur: Int)

fun main(args: Array<String>) {
    val mhs = Mahasiswa("Susi Susanti", 23)
    val mhs2 = Mahasiswa("Agus Budianto", 25)
    println("Nama Mahasiswa: ${mhs.nama}")
    println("Umur Mahasisa:  ${mhs.umur}")
    println("Nama Mahasiswa: ${mhs2.nama}")
    println("Umur Mahasisa:  ${mhs2.umur}")
}
```

17. Modifikasilah program di atas sehingga menjadi seperti berikut, dan perhatikan hasil running program.

```
data class Mahasiswa(val nama: String, val umur: Int)

fun main(args: Array<String>) {
    val mhs = Mahasiswa("Susi Susanti", 23)
    val mhs2 = Mahasiswa("Susi Susanti", 23)
    val mhs3 = Mahasiswa("Agus Budianto", 25)

    if (mhs.equals(mhs2) == true)
        println("mhs sama dengan mhs2.")
    else
        println("mhs tidak sama dengan mhs2.")

    if (mhs.equals(mhs3) == true)
        println("mhs sama dengan mhs3.")
    else
        println("mhs tidak sama dengan mhs3.")

    println("Hashcode dari mhs: ${mhs.hashCode()}")
    println("Hashcode dari mhs2: ${mhs2.hashCode()}")
    println("Hashcode dari mhs3: ${mhs3.hashCode()}")
}
```

18. Buat program seperti berikut, dan perhatikan hasil running program.

```
data class Mahasiswa(val nama: String, val umur: Int)

fun main(args: Array<String>) {
    val mhs = Mahasiswa("Susi Susanti", 23)

    // mengkopi umur dari objek mhs
    val mhs2 = mhs.copy(nama = "Lusiana")

    println("Nama ${mhs.nama}, Umur ${mhs.umur}")
    println("Nama ${mhs2.nama}, Umur ${mhs2.umur}")
}
```

19. Tambahlkan program di atas dengan kode berikut pada bagian paling bawah.

```
    val nama = mhs.component1()
    val umur = mhs.component2()
    println("Nama $nama, Umur $umur")
```

LATIHAN

1. Tulis fungsi untuk menghitung jarak dua buah titik t1(x1, y1) dan t2(x2, y2). Panggilah fungsi ini dalam fungsi main() dengan t1(2,3) dan t2(8,7), serta t1(5,3) dan t2(-8, -4).
2. Buatlah fungsi inline dengan nama **pangkat**, dengan contoh pemanggilan seperti berikut.

```
println("4 pangkat 3 = ${pangkat(4,3)}")
```

3. Buat kelas Barang dengan properti: nama, harga, jumlah, dan diskon. Berilah nilai default pada konstruktornya. Tambahkan fungsi(method):
    - tampil() → untuk menampilkan semua propertinya
    - hitungTotal() →untuk menghitung harga x jumlah x diskon

    Buatlah fungsi main() untuk mengakses kelas Barang

TUGAS

1. Tuliskan soal tugas yang harus dikerjakan mahasiswa di

REFERENSI

1. https://play.kotlinlang.org/byExample/overview
2. https://www.guru99.com/kotlin-tutorial.html
3. https://codelabs.developers.google.com/android-kotlin-fundamentals/
4. https://beginnersbook.com/2017/12/kotlin-tutorial/