# LAPORAN TUGAS

## BAHASA PEMROGRAMAN 3

### **(*Dosen : Dede Husen, M.Kom.*)**

Modul 1



**Nama : Muhammad Rizal Nurfirdaus**

**NIM : 20230810088**

**Kelas : TINFC-2023-04**

# TEKNIK INFORMATIKA

## FAKULTAS ILMU KOMPUTER

**UNIVERSITAS KUNINGAN**

**PRETEST**

1. Apa perbedaan antara Kotlin dengan Java?

Kotlin adalah bahasa pemrograman modern yang dirancang untuk lebih ringkas, aman, dan mudah dibaca dibandingkan Java. Kotlin berjalan di atas JVM (Java Virtual Machine) sama seperti Java, tetapi memiliki sintaks yang lebih sederhana dan fitur tambahan seperti null safety, extension functions, dan coroutines, sementara Java masih cenderung verbose (panjang dan berulang dalam penulisan kode).

2. Apa yang dimaksud dengan null safety di Kotlin?

Null safety adalah fitur di Kotlin yang mencegah terjadinya *NullPointerException* (NPE). Kotlin membedakan antara nullable (String?) dan non-nullable (String) variabel. Artinya, variabel tidak bisa menyimpan null kecuali secara eksplisit diizinkan, sehingga mengurangi potensi error saat runtime.

3. Apa kelebihan utama Kotlin dibandingkan dengan Java, dan mengapa Kotlin dipilih sebagai bahasa utama untuk pengembangan aplikasi Android?

Kelebihan utama Kotlin adalah:

* Sintaks lebih ringkas dan ekspresif, mengurangi jumlah kode boilerplate.
* Mendukung null safety, sehingga lebih aman dari error.
* Mendukung coroutines untuk pengelolaan proses asynchronous yang efisien.
* Interoperable dengan Java bisa menggunakan kode Java secara langsung.

Kotlin dipilih sebagai bahasa utama Android karena meningkatkan produktivitas developer, keamanan kode, dan kompatibilitas penuh dengan Java serta Android Studio.

**PRAKTIKUM**

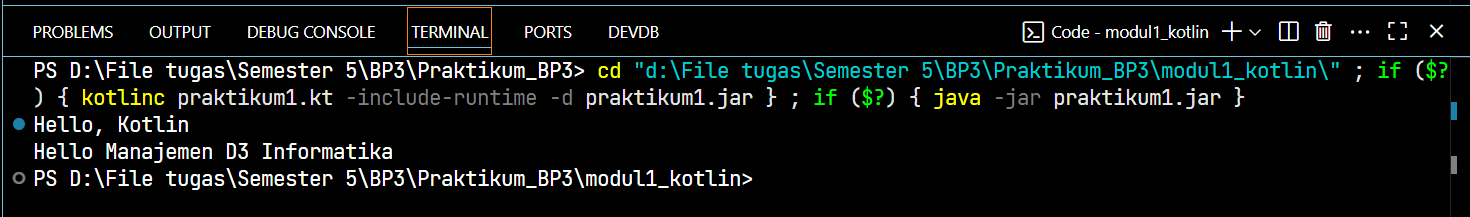
Praktikum 1 – Basic Sintak  
Source Code:

fun main ( ) {

println ("Hello, Kotlin")

println ("Hello Manajemen D3 Informatika")

}

Hasil Run:   


Praktikum 2 – Variabel

Source Code:

// Praktikum2.kt

fun main () {

// penggunaan keyword var mutable (bisa diubah)

var nama : String = "Dede" // contoh penulisan dengan tipe data

var asal = "Kuningan" // contoh penulisan tanpa tipe data

var usia = 25

// nilai variable yang diubah

nama = "Dede Husen"

asal = "Kuningan Jawa Barat"

usia = 30

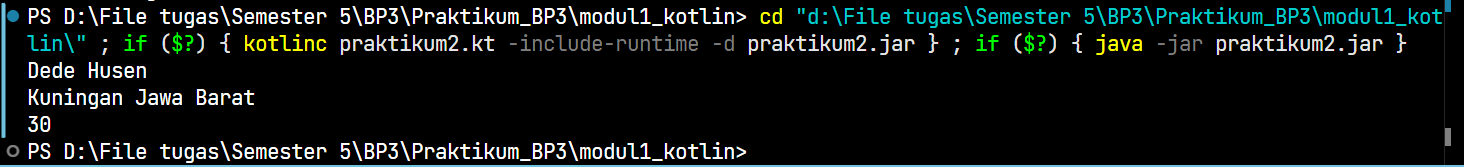
println (nama)

println (asal)

println (usia)

}

Hasil Run:



Praktikum 3 – Fungsi

Source Code:

// Praktikum3.kt

// fungsi standar

fun ucapkan\_salam(){

    println("Assalamu'alaikum...selamat siang warga fkom semuanya")

}

// fungsi dengan argument

fun salam(nama: String) {

    println("hallo Nama saya, $nama")

}

// fungsi dengan nilai return

fun kali(a:Int, b:Int): Int {

    return a\*b

}

fun tambah(a:Int, b:Int): Int = a+b

fun bagi(a:Float, b:Float) : Unit {

    println ("Pembagian antara $a dan $b adalah ${a/b}")

}

fun kurang(a:Int, b:Int){

    println("Pengurangan antara $a dan $b adalah ${a-b}")

}

fun main () {

    println("======Fungsi standar kotlin======")

    ucapkan\_salam() // cara memanggil fungsi

    salam("Dede")

    print("hasil perkalian antara 3 dan 10 adalah = ")

    println(kali(3,10))

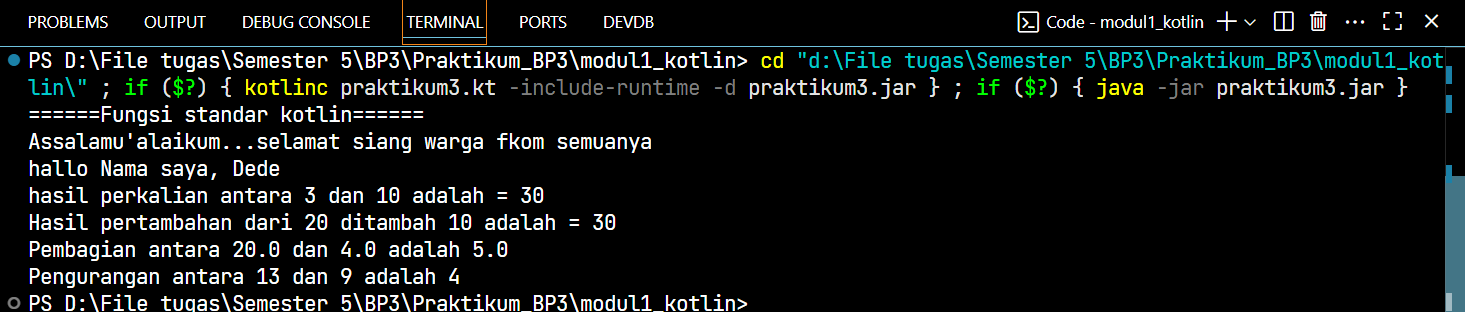
    println("Hasil pertambahan dari 20 ditambah 10 adalah = ${tambah(20,10)}")

    bagi(20f, 4f)

    kurang(13,9)

}

Hasil Run:



Praktikum 4 – IF Else Statement (Conditional Statement)

Source Code:

// Praktikum4.kt

fun main () {

    cek\_login()

}

fun cek\_login(){

    val username = "admin"

    val password = "FkomJuara1"

    if (username == "admin" && password == "FkomJuara1"){

        println("Anda berhasil masuk")

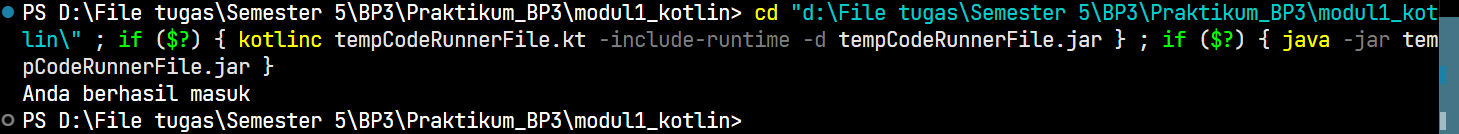
    } else {

        println("Kombinasi username dan password Anda salah")

    }

}

Hasil Run:



Praktikum 5 – When Expression

Source Code:

fun main () {

    cek\_nama\_hari()

    cek\_khodam()

}

//fungsi cek\_nama\_hari()

fun cek\_nama\_hari() {

    val hari = 4 // input nomor hari

    val namaHari = when (hari) {

        1 -> "Senin"

        2 -> "Selasa"

        3 -> "Rabu"

        4 -> "Kamis"

        5 -> "Jum'at"

        6 -> "Sabtu"

        7 -> "Minggu"

        else -> "Input hari salah"

    }

    println("Hari ini adalah hari $namaHari")

}

//fungsi cek khodam

fun cek\_khodam () {

    val khodam = "Januari"

    when (khodam) {

        "Januari", "Februari", "Maret" -> println("Khodam Anda adalah Harimau")

        "April", "Mei", "Juni", "Juli" -> println("Khodam Anda adala Kucing")

        "Agustus", "September", "Oktober" -> println("Khodam Anda adala Macan Ciremai")

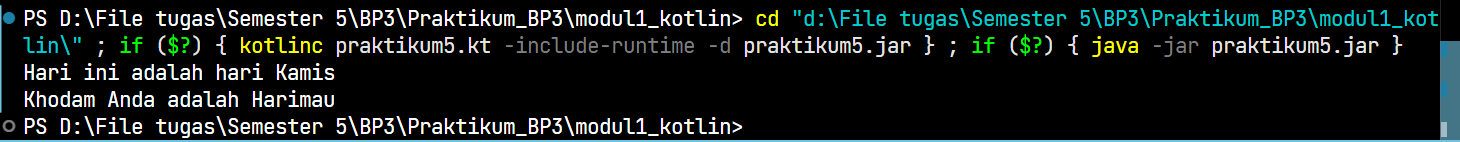
        "November", "Desember" -> println("Khodam Anda adala Ratu Pantai selatan")

        else -> println("Anda tidak punya khodam")

    }

}

Hasil Run:



Praktikum 6 – Looping ( for, while dan do-while)

Source Code:

fun main () {

    println("======Looping for======")

    cetak\_angka\_for()

    println("======Looping while======")

    cetak\_angka\_while()

    println("======Looping do-while======")

    cetak\_angka\_do\_while()

}

fun cetak\_angka\_for () { //for

    // var i=1 // This line is commented out in the image and not needed for a range-based for loop.

    println("=====Print angka 1-5 dengan for=====")

    for (i in 1..5) {

        println(i)

    }

}

fun cetak\_angka\_while () { //while for

    var i=1

    println("=====print angka dengan while 1 sampai 5")

    while(i<=5){

        println(i)

        i++

    }

}

fun cetak\_angka\_do\_while () { // do while

    var i=1

    println("=====Print angka dengan do while 1 sampai 5")

    do {

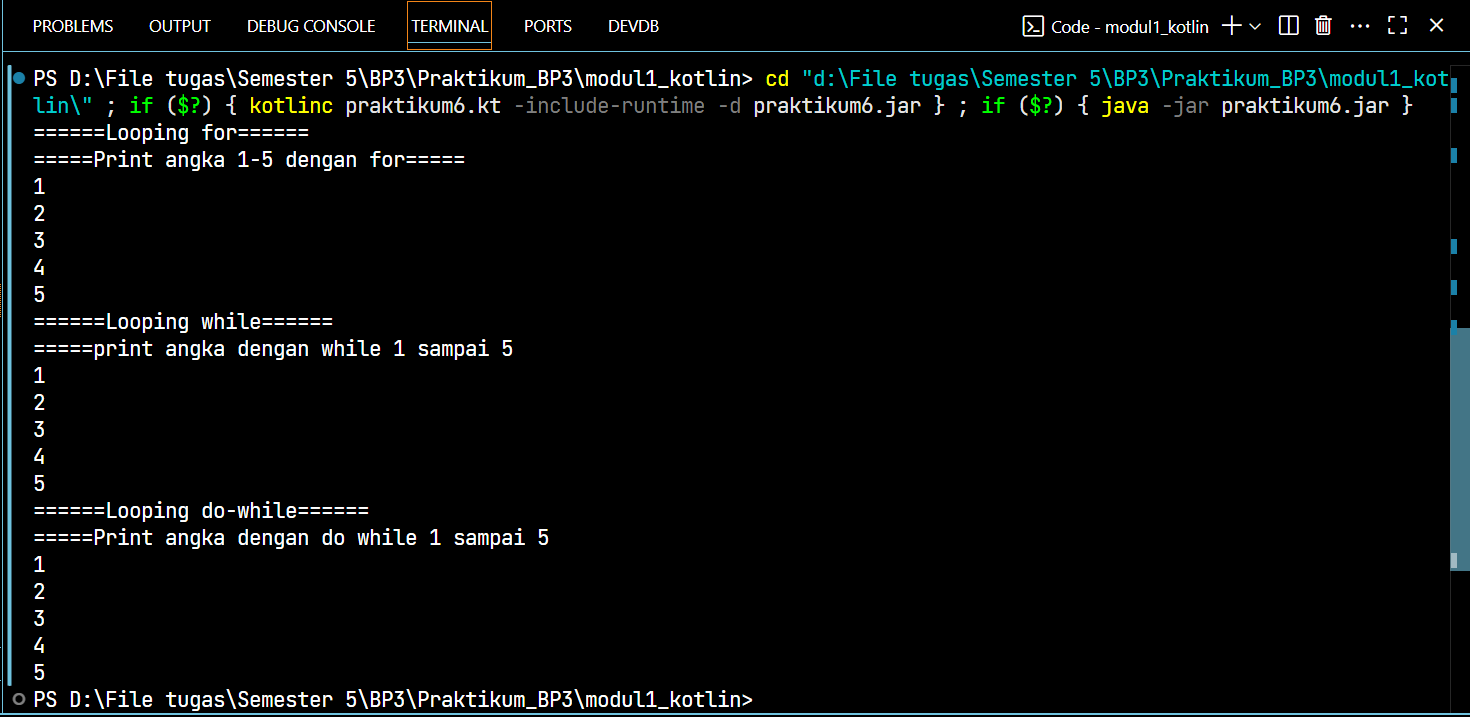
        println(i)

        i++

    } while(i<=5)

}

Hasil Run:



**POSTTEST**

1. Dari praktikum di atas, apakah variabel dapat dituliskan tanpa menyebutkan tipe datanya? Jelaskan.

Ya, variabel dapat dituliskan tanpa menyebutkan tipe datanya dalam Kotlin.  
Hal ini dimungkinkan karena adanya fitur Type Inference (inferensi tipe). Ketika variabel dideklarasikan menggunakan kata kunci var atau val dan langsung diberikan nilai, kompiler Kotlin secara otomatis menentukan tipe data variabel tersebut berdasarkan tipe nilai yang diinisialisasikan.

Contoh (dari Praktikum 2):

var asal = "Kuningan" // tipe data diinferensi sebagai String

var usia = 25 // tipe data diinferensi sebagai Int

1. Setelah mempraktikkan praktikum di atas, apa saja yang membuat Kotlin lebih sederhana dari Java?

Beberapa hal yang membuat Kotlin lebih sederhana dan ringkas dibandingkan Java, antara lain:

* Type Inference: Kotlin tidak memerlukan penulisan tipe data secara eksplisit, sehingga mengurangi kode berulang (*boilerplate code*).
* Fungsi dengan Expression Body: Fungsi sederhana yang hanya mengembalikan satu nilai dapat ditulis dalam satu baris.
* fun tambah(a: Int, b: Int): Int = a + b Penulisan ini jauh lebih ringkas dibandingkan Java.
* Pernyataan when: Sebagai pengganti switch di Java, when lebih fleksibel karena dapat digunakan sebagai *expression*, mendukung rentang nilai (*range*), kombinasi kondisi, dan pengecekan tipe data.
* Sintaks Lebih Ringkas: Deklarasi variabel, fungsi, dan loop di Kotlin umumnya membutuhkan lebih sedikit kode dan modifier dibandingkan Java.

1. Pada praktikum ke-6 terdapat script for (i in 1..5), jelaskan kegunaan kode tersebut.

Kode for (i in 1..5) digunakan untuk melakukan perulangan (iterasi) pada rentang angka dari 1 hingga 5.

Penjelasan tiap bagiannya:

* for → Menandai awal struktur perulangan.
* i → Variabel iterasi yang menyimpan nilai setiap kali perulangan berjalan.
* in → Operator yang digunakan untuk mengiterasi elemen dari suatu koleksi atau rentang.
* 1..5 → Operator range (..) yang mendefinisikan rentang inklusif dari 1 sampai 5.

Artinya, blok kode di dalam perulangan tersebut akan dijalankan sebanyak lima kali, dengan nilai i berturut-turut adalah 1, 2, 3, 4, dan 5.

**LATIHAN/TUGAS**

1. Buatlah sebuah program sederhana yang dapat menghitung nilai ujian siswa dengan rentang nilai sebagai berikut:  
   a) Jika nilai siswa 80 – 100 = Nilai A  
   b) Jika nilai siswa 60 – 79 = Nilai B  
   c) Jika nilai siswa 50 – 59 = Nilai C  
   d) Jika nilai kurang dari 50 = Nilai D
2. Buat dengan menggunakan referensi baik dari modul maupun dari luar.
3. Silahkan submit di hybrid hasil pekerjaannya.

Source Code:

fun main() {

    // Contoh nilai ujian siswa

    val nilaiUjian = 85

    // Memanggil fungsi untuk mendapatkan nilai huruf

    val nilaiHuruf = hitungNilai(nilaiUjian)

    // Menampilkan hasil

    println("Nilai angka siswa: $nilaiUjian")

    println("Nilai huruf yang didapat: $nilaiHuruf")

}

/\*\*

 \* Fungsi untuk menghitung nilai huruf berdasarkan rentang nilai angka.

 \* Menggunakan ekspresi 'when' (sebagai pengganti switch yang lebih canggih di Kotlin).

 \*

 \* @param nilai Angka ujian siswa (diasumsikan 0-100).

 \* @return Nilai huruf (A, B, C, atau D).

 \*/

fun hitungNilai(nilai: Int): String {

    // Ekspresi 'when' digunakan untuk mencocokkan nilai dengan rentang

    return when (nilai) {

        // a) Jika nilai siswa 80 - 100 = Nilai A

        in 80..100 -> "Nilai A"

        // b) Jika nilai siswa 60 - 79 = Nilai B

        in 60..79 -> "Nilai B"

        // c) Jika nilai siswa 50 - 59 = Nilai C

        in 50..59 -> "Nilai C"

        // d) Jika nilai kurang dari 50 = Nilai D

        in 0..49 -> "Nilai D"

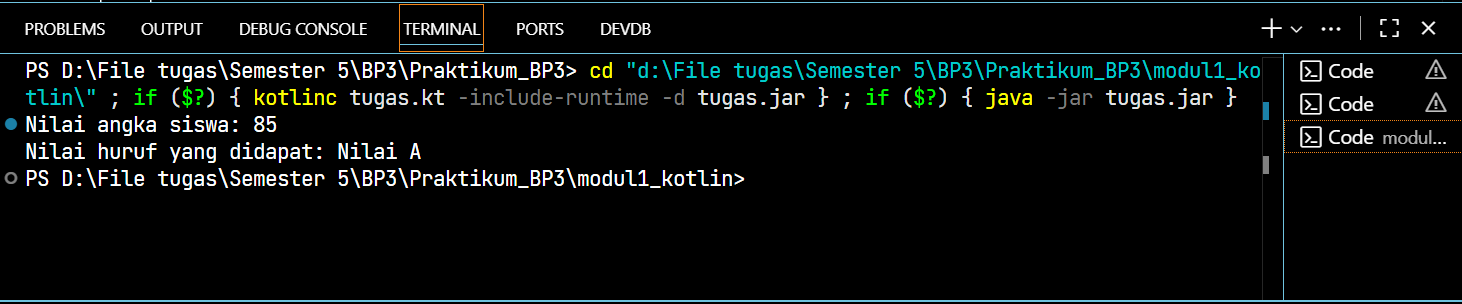
        // Kasus jika nilai di luar rentang 0-100 (opsional)

        else -> "Nilai Tidak Valid"

    }

}

Hasil Run:



Analisis: Program Kotlin ini menunjukkan implementasi kontrol alir yang **efisien** dan **ringkas** menggunakan fungsi when sebagai ekspresi untuk menentukan nilai huruf berdasarkan rentang nilai angka. Fungsi hitungNilai(nilai: Int): String adalah inti program, yang mengambil nilai ujian bertipe Int dan mengembalikan nilai huruf bertipe String. Penggunaan when menghilangkan kebutuhan akan serangkaian if-else if yang panjang dan *verbose* (bertele-tele) seperti di bahasa pemrograman lain.

Fitur utama yang ditekankan adalah penggunaan operator **Range Inklusif (..)** bersama kata kunci **in**, yang memungkinkan pengecekan kondisi secara elegan. Misalnya, in 80..100 -> "Nilai A" secara langsung menyatakan bahwa jika nilai berada di antara 80 dan 100 (termasuk kedua batas), hasilnya adalah "Nilai A". Karena when digunakan sebagai ekspresi, kode dapat langsung mengembalikan nilai tanpa variabel sementara. Hal ini menjadikan kode **bersih**, **mudah dibaca**, dan memenuhi prinsip Kotlin tentang **keringkasan** (*conciseness*), sekaligus memastikan semua kemungkinan kondisi dicakup dengan adanya blok else untuk menangani nilai yang tidak valid (di luar rentang 0-100).