DASAR PEMROGRAMAN KOTLIN

Modul 15: Penanganan Eksepsi & Pengenalan Coroutines

Disusun Oleh:
Kusuma Web Academy

Mengamankan Kegagalan Program: Try-Catch-Finally

Kesalahan dinamis di waktu runtime (*runtime exceptions*) wajib dikelola secara anggun agar tidak menghentikan jalannya aplikasi secara sepihak.

- **try Block:** Membungkus baris perintah kode program yang memiliki risiko kegagalan sistem.
- **catch Block:** Menangkap galat eksepsi spesifik, lalu menjalankan logika pemulihan alternatif yang aman.
- **finally Block:** Blok pembersihan memori atau penutupan jaringan eksternal yang **pasti** dieksekusi, apa pun hasil akhir try-catch sebelumnya.

Pengenalan Pemrograman Asinkron: Coroutines

Kotlin Coroutines memungkinkan eksekusi proses multitasking yang sangat berat secara non-blocking dengan konsumsi memori yang super hemat.

- **Multitasking Ringan:**
  - Jauh lebih efisien daripada thread sistem biasa. Satu thread sistem dapat mengelola ribuan coroutine secara bersamaan tanpa kehabisan memori.
- **suspend Functions:**
  - Ditandai dengan kata kunci `suspend`. Fungsi asinkron khusus ini diizinkan untuk menjeda eksekusinya sementara waktu tanpa memblokir thread antarmuka utama (*Main Thread UI*).

Praktik Penanganan Eksepsi Mulus

Kode implementasi penanganan eksepsi pembagian nol di laboratorium komputer:

```
fun main() {
    val angkaAwal = 50
    val pembagi = 0

    try {
        // Baris berisiko memicu ArithmeticException
        val hasilBagi = angkaAwal / pembagi
        println("Hasil Pembagian: $hasilBagi")
    } catch (e: ArithmeticException) {
        println("Sistem Mengamankan Galat: Pembagian dengan nol
            dilarang keras!")
    } finally {
        println("Proses audit transaksi materi Kusuma Web ditutup
            dengan aman.")
    }
}
```