**Rangkuman dari Kotlin Functional Programming**

Sebelum lanjut ke paradigma Object-Oriented Programming, mari kita rangkum apa saja yang sudah Anda pelajari pada sub-modul Functional Programming.

* Anatomi dari sebuah function terdiri dari 2 (adua) bagian utama, yaitu function header dan function body:
  + **Function Header**  
    Function header adalah bagian yang merupakan konstruksi dari sebuah fungsi untuk menentukan perilakunya akan seperti apa. Di dalam function header terdapat visibility modifier, kata kunci fun, nama, daftar parameter dan nilai kembalian dari fungsi tersebut.
  + **Function Body**  
    Function Body adalah bagian yang dalamnya kita akan menempatkan sebuah logika kode baik itu sebuah expression atau statement.
* Kotlin memiliki fitur seperti named dan default argument yang dapat menghindari kita dari kesalahan saat menyematkan sebuah argumen saat menggunakan sebuah fungsi.
* Selain named dan default argument, Kotlin juga memiliki fitur varargs yang dapat digunakan untuk menyederhanakan deklarasi beberapa parameter yang memiliki tipe yang sama.
* Kotlin mendukung 2 (dua) extension yang dapat digunakan, yaitu *Extension Functions* dan *Extension Properties*. Jika extension functions digunakan untuk menambahkan fungsi baru, extension properties tentunya digunakan untuk menambahkan sebuah properti baru.
* Lambda expression, atau biasa disebut dengan anonymous function atau function literal adalah fitur yang cukup populer sampai sekarang dalam dunia functional programming. Bisa disebut sebagai anonymous karena lambda tidak memiliki sebuah nama seperti halnya sebuah fungsi pada umumnya. Karena merupakan sebuah fungsi, lambda juga dapat memiliki daftar parameter, body dan return type. Berikut adalah beberapa karakteristik dari Lambda pada kotlin.
  + Dalam menggunakan lambda, kita tidak perlu mendeklarasi tipe spesifik untuk nilai kembaliannya. Tipe tersebut akan ditentukan oleh kompiler secara otomatis.
  + Walaupun merupakan sebuah fungsi, lambda tidak membutuhkan kata kunci fun dan *visibility modifier* saat dideklarasikan, karena lambda bersifat anonymous.
  + Parameter yang akan ditetapkan berada di dalam kurung kurawal {}.
  + Ketika ingin mengembalikan nilai, kata kunci return tidak diperlukan lagi karena kompiler akan secara otomatis mengembalikan nilai dari dalam body.
  + Lambda expression dapat digunakan sebagai argumen untuk sebuah parameter dan dapat disimpan ke dalam sebuah variabel.
* *Higher-Order Function*, yaitu sebuah fungsi yang menggunakan fungsi lainnya sebagai parameter, menjadikan tipe kembalian, ataupun keduanya. High-order function adalah salah satu fitur yang memanfaatkan Lambda.
* Kotlin standard library memiliki beberapa fungsi yang tujuan utamanya adalah bagaimana menuliskan logika kode di dalam konteks dari sebuah objek. Dalam menggunakan fungsi tersebut kita akan memanfaatkan lambda expression yang memiliki ruang lingkupnya sendiri, di mana dalam ruang lingkup tersebut kita dapat mengakses konteks dari sebuah objek. Fungsi inilah yang dinamakan sebagai scope function. Beberapa fungsi yang berada di dalamnya antara lain, let, run, with, apply, dan also.
* Sama seperti Java, Kotlin  mendukung juga mekanisme reflection yang berarti seperangkat fitur bahasa dan library yang memungkinkan kita untuk mengamati struktur kode dari proyek yang sedang kita kerjakan secara langsung.
* Saat menggunakan Kotlin, kita bisa mendeklarasi fungsi di dalam sebuah fungsi dan menggunakannya hanya di  dalam fungsi tersebut.
* Kotlin Collection adalah salah satu struktur data mumpuni yang banyak menyediakan fungsi untuk memudahkan kita dalam mengelola dan memanipulasi data. Pada sub-modul sebelumnya, kita sudah mempelajari beberapa fungsi yang disediakan seperti **map()**, **sum()**, dan **sorted()**. Selain beberapa fungsi yang sudah disebutkan, berikut fungsi lain yang dapat kita manfaatkan:
  + **Fold**  
    Fold, Anda bisa dengan mudah melakukan perhitungan setiap nilai yang berada di dalam sebuah collection tanpa harus melakukan iterasi item tersebut satu-persatu menggunakan fungsi **fold()**.
  + **Drop**  
    **drop()**, fungsi yang bisa kita manfaatkan untuk memangkas item yang berada di dalam sebuah objek collection berdasarkan jumlah yang kita tentukan.
  + **Take**  
    Fungsi **take()** bisa kita manfaatkan untuk menyaring item yang berada di dalam sebuah objek collection.
  + **Slice**  
    Bagaimana jika kita ingin menyaring item dari posisi tertentu? Untuk itu Anda bisa memanfaatkan fungsi **slice()**.
  + **Distinct**  
    Saat berurusan dengan item yang sama di dalam sebuah collection, untuk menyaring item yang sama tersebut kita akan melakukan iterasi dan membandingkan setiap itemnya. Namun dengan Kotlin kita tidak perlu melakukannya secara manual, karena Kotlin Collection menyediakan fungsi untuk melakukannya dengan mudah yaitu fungsi **distinct()**.
  + **Chunked**  
    Sama seperti fungsi **split()**, fungsi **chunked()** bisa kita gunakan untuk memecah nilai String menjadi beberapa bagian kecil dalam bentuk Array.
* Recursion merupakan sebuah teknik dasar dalam pemrograman yang bisa kita gunakan untuk menyederhanakan pemecahan masalah yang umumnya diselesaikan dengan cara yang kompleks. Di Kotlin, recursion disebut juga dengan *recursive function*.