**Rangkuman dari Kotlin Generics**

Inti yang telah dipelajari pada modul ini adalah *Generics*, sebuah konsep yang memungkinkan suatu kelas atau interface menjadi tipe parameter yang dapat digunakan untuk berbagai macam tipe data. Tidak hanya itu saja, kita telah mengenal juga tentang *variance*, yaitu salah satu konsep dari *generics*. Berikut adalah rangkuman dari keseluruhan materi tersebut :

* *Generics* pada kotlin dapat diterapkan dengan meletakkan tipe parameter ke dalam angle brackets (**<>**). Tipe parameter atau alias ini merupakan word yang bisa bebas Anda deklarasikan, contoh yang paling umum adalah menggunakan word  ‘**T**’.

1. interface List<T>{
2. operator fun get(index: Int) : T
3. }

* Ada 2 jenis penerapan generics, yaitu **Generics pada Class**, dan **Generics pada Function**.
* **Generics pada Class** dilakukan dengan meletakkan tipe parameter setelah penulisan nama class.

1. interface List<T>{
2. operator fun get(index: Int) : T
3. }

* Class yang mewarisi sebuah **Generic Class** atau Interface harus menentukan tipe argumen sebagai tipe dasar dari parameter generic kelas tersebut. Tipe parameter yang digunakan bisa berupa tipe yang spesifik (Integer, Float, Double, atau Object lainnya) atau bisa juga class tersebut mendeklarasikan tipe parameter yang lainnya.
* **Generics pada Function** dilakukan dengan meletakkan tipe parameter sebelum penulisan nama dari fungsi tersebut.

1. fun <T> run(): T {
2. /\*...\*/
3. }

* Dalam menggunakan sebuah class maupun function generic, tipe parameter diletakkan setelah memanggil nama dari sebuah function ataupun class.

1. fun main() {
2. val longArrayList = ArrayList<Long>()
3. displayArray<Long>(longArrayList)
4. }

* *Constraint Type Parameter* adalah teknik untuk membatasi cakupan tipe data apa saja yang dapat di sediakan oleh **Generic Class** maupun **Generic Function**. Pembatasan ini dilakukan dengan membubuhkan tanda *colon* (:) setelah tipe parameter yang kemudian diikuti oleh tipe data yang akan dijadikan batasan.
* Batasan dalam **Constraint Type Parameter** adalah*child class* dari tipe data batasan tersebut. Jadi, semisal ditentukan batasannya adalah tipe **List**, maka hanya berlaku pada *child class* dari class tersebut.
* **Variance**, adalah konsep pada Generics yang menggambarkan bagaimana sebuah tipe yang memiliki subtipe yang sama dan tipe argumen yang berbeda saling berkaitan satu sama lain. Ada beberapa jenis dari *variance* yaitu:
  + **Covariant**  
    Ditandai dengan *keyword* ‘**out**’ sebelum deklarasi dari tipe parameter. Nilai dari tipe parameter tersebut hanya bisa diproduksi seperti menjadikanya sebagai return type dan tidak dapat dikonsumsi seperti menjadikannya sebagai tipe argumen untuk setiap fungsi di dalam kelas tersebut.
  + **Contravariant**  
    Ditandai dengan *keyword* ‘**in**’ sebelum deklarasi dari tipe parameter. Nilai dari tipe parameter tersebut hanya bisa dikonsumsi seperti menjadikannya sebagai tipe argumen untuk setiap fungsi dan tidak untuk dikonsumsi sebagai *return type* dari sebuah function di dalam kelas tersebut