Generic & Multi-Threading

Introduction Generic Programming

1. Apa itu generic?

Generic programming adalah salah satu cara pembuatan class yang dapat digunakan untuk semua jenis tipe data (konsep yang memungkin class atau interface menjadi tipe parameter), sehingga penggunaannya lebih fleksibel dan tidak kaku terhadap satu jenis tipe data saja.

* Semua generic method di deklarasikan dengan kurung tajam (< E >).

1. Contoh generic
   1. Non-generic parameter

ArrayList list = **new** ArrayList();

list.add("binus");

String str = (String) list.get(0);

* 1. Generic Parameters

ArrayList<String> list = **new** ArrayList<String>();

list.add("Binus Alam Sutera");

String str = list.get(0); // ga perlu casting

1. Generic type 1

**public** **class** Box {

**private** Object object;

**public** **void** set(Object object) { **this**.object = object; }

**public** Object get() { **return** object; }

}

**class** name<T1, T2, ..., Tn> { /\* ... \*/ }

**public** **class** Box<T> {

// T stands for "Type"

**private** T t;

**public** **void** set(T t) { **this**.t = t; }

**public** T get() { **return** t; }

}

1. Generic type 2

* Objek dalam contoh tersebut dapat digantikan oleh T
* Jenis variabel, parameter dapat berupa tipe non-primitif. Contohnya, Kelas, interface, array, atau bahkan variabel tipe lainnya

**public** **class** BoxInTheBox<T,Box<String>> {

/\* Some code \*/

}

1. Yok buat sama-sama
   1. Buat project baru pada eclipse beserta class yang akan digunakan.
   2. Pada contoh kali ini saya akan membuat 2 class;
      * Generic class (tanpa method main) -> buat class ini terlebih dahulu
      * Main Class
   3. Buat generic class, lalu tambahkan encapsulation seperti contoh dibawah ini

**class** genericSample<T> {

//T Adalah Kepanjangan dari Tipe Parameter

**private** T Manusia;

//Method Void Set Dengan Tipe Parameter T

**protected** **void** setManusia(T Manusia){

**this**.Manusia = Manusia;

}

//Method Return Get Yang Mengembalikan Nilai Manusia

**protected** T getManusia(){

**return** Manusia;

}

}

\*catatan:

- < T > berarti parameter yang digunakan untuk menentukan jenis tipe data yang akan digunakan pada variabel atau atribut di dalamnya.

- T bisa digantikan dengan tipe daya apa saja (Integer, String, Boolean, dsb)

- generic class hanya menerima tipe data non-primitive

d. Membuat Main Class

**public** **class** latihan\_java {

**public** **static** **void** main(String[] args){

//Membuat Instance dan Mendefinisikan Tipe Data String

GenericClass<String> nama = **new** GenericClass<>();

nama.setManusia("Ricko");

System.***out***.println("Nama Saya: "+nama.getManusia());

//Membuat Instance dan Mendefinisikan Tipe Data Integer

GenericClass<Integer> umur = **new** GenericClass<>();

umur.setManusia(19);

System.***out***.println("Usia Saya: "+umur.getManusia()+" Tahun");

//Membuat Instance dan Mendefinisikan Tipe Data Boolean

GenericClass<Boolean> kondisi = **new** GenericClass<>();

kondisi.setManusia(**true**);

System.***out***.println("Saya Sedang Belajar Generic Java: "+kondisi.getManusia());

//Membuat Instance dan Mendefinisikan Tipe Data Float

GenericClass<Float> IPK = **new** GenericClass<>();

IPK.setManusia(3.98f);

System.***out***.println("Nilai IPK Saya: "+IPK.getManusia());

//Membuat Instance dan Mendefinisikan Tipe Data Character

GenericClass<Character> Karakter = **new** GenericClass<>();

Karakter.setManusia('R');

System.***out***.println("Huruf Awalan Nama Saya Adalah: "+Karakter.getManusia());

}

}

Referensi yang bisa kalian buka:

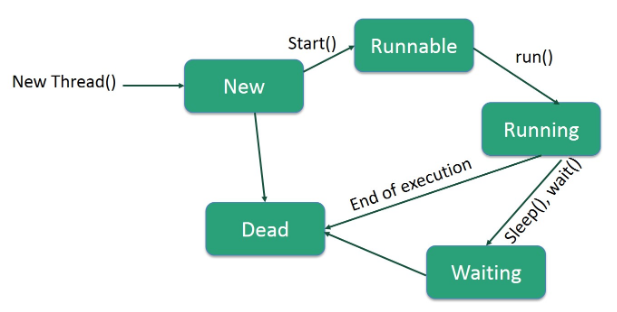
<https://codeandwork.github.io/courses/java/generics_and_threads.html>

Introduction Multi-Threading

1. Apa itu Multi-Threading?

Multi-Threading berarti kita dapat membangun multi-threaded program dengan menggunakan Java. Multi-threaded program memiliki dua bagian atau lebih yang dapat di run secara bersamaan dan tiap bagiannya dapat menangani tugas(task) yang berbeda di waktu yang bersamaan.

1. Life Cycle of a Thread



New -> thread baru mulai life cycle nya (born)

Runnable -> Setelah lahir, thread nya menjadi runnable. Dalam kondisi ini dianggap menjalankan tugasnya.

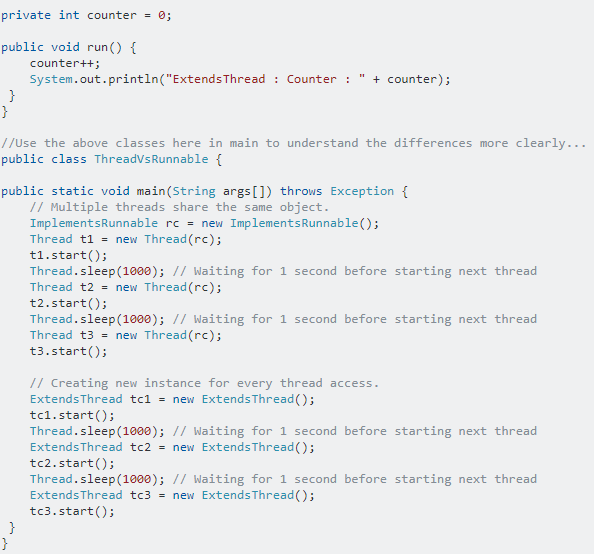
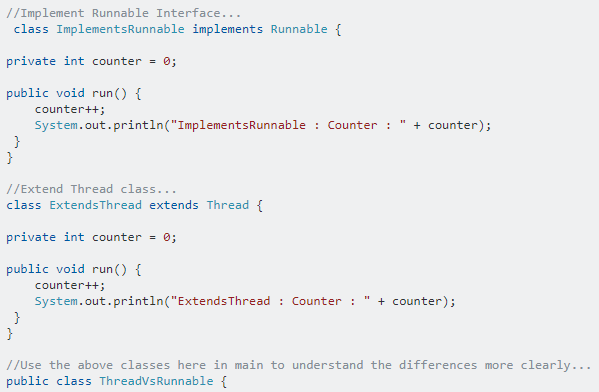
Waiting -> Terkadang, transisi thread harus menunggu menunggu, sementara thread menunggu thread lain untuk melakukan tugas. Thread transisi kembali ke status runnable hanya ketika thread lain memberi sinyal thread menunggu untuk melanjutkan eksekusi.

1. “Implements Runnable” vs “extends Thread”

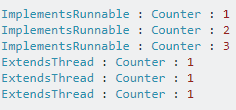
Thread dapat dibuat dengan 2 cara; dengan extends Thread (class) atau dengan implement Runnable (interface)

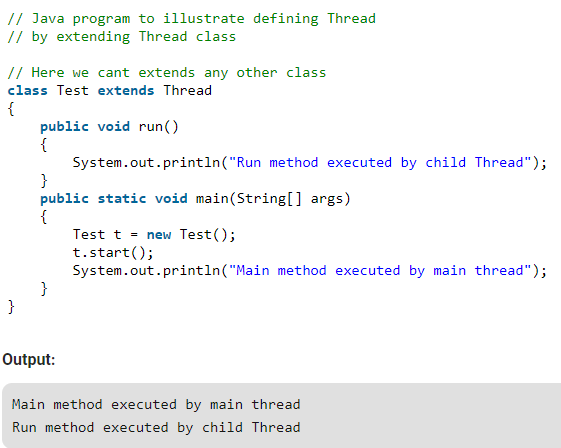
Perbedaan yang dapat dilihat dari keduanya yaitu:

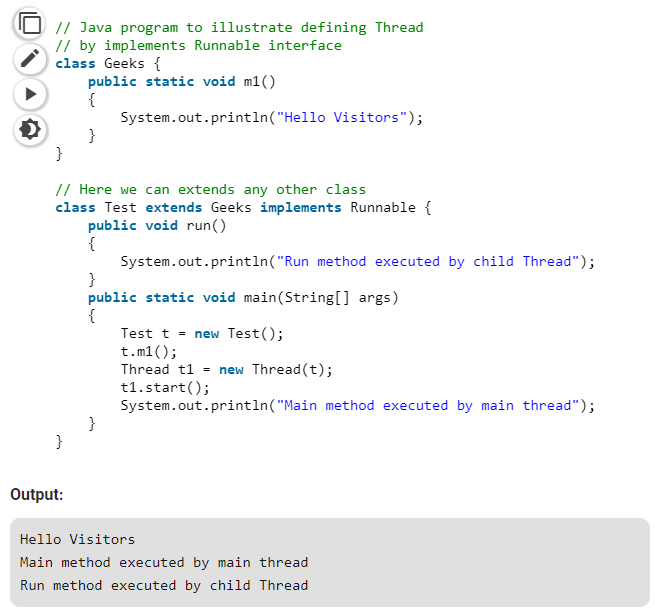
* Pada saat extends Thread class, kita tidak dapat neg-extends kelas lain, bahkan keals lain yang kita butuhkan.
* Sedangkan jika kita meng-implement Runnable, kita dapat menyimpan tempat untuk meng-extends kelas lain jika diperlukan.
* Pada saat extends Thread Class, tiap thread akan membuat objek unik dan mengaitkannya dengan hal unik tersebut.
* Sedangkan pada saat menggunakan implement Runnable, akan dibagikan objek yang sama kepada multiple threads.



Output dari program diatas







Contoh lain silahkan lihat pada kodingan yang diberikan

1. Kapan harus menggunakan implement Runnable?

* Ketika mau mengakses resource yang sama dari beberapa threads
* Hindari menggunakan Thread class (karena multiple object mengkonsumsi memory lebih banyak)
* Kelas yang mengimplementasikan Runnable bukan thread dan hanya class. Agar Runnable menjadi Thread, anda hatus membuat instance Thread dan passing menjadi target.
* Ketika ada kebutuhan untuk extends superclass, meng-implement Runnable interface lebih tepat dari pada meng-extends Thread Class.

Silahkan lihat referensi lainnya:

* <https://www.tutorialspoint.com/java/java_multithreading.htm>
* <https://stackoverflow.com/questions/541487/implements-runnable-vs-extends-thread-in-java>
* <https://www.geeksforgeeks.org/implement-runnable-vs-extend-thread-in-java/#:~:text=The%20significant%20differences%20between%20extending%20Thread%20class%20and%20implementing%20Runnable%20interface%3A&text=When%20we%20extend%20Thread%20class%2C%20each%20of%20our%20thread%20creates,same%20object%20to%20multiple%20threads.>

Terimakasih 😊