MODUL 4
Multi Threading 1

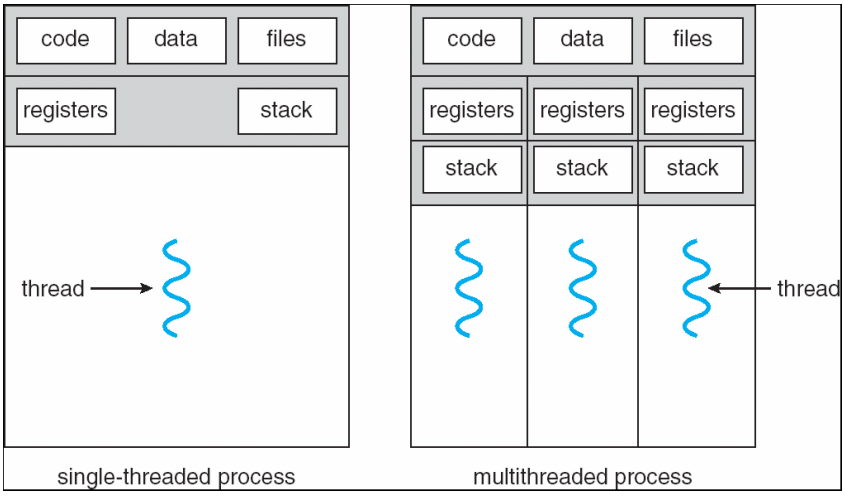
A. Tujuan :

1. Memahami tentang single thread
2. Memahami konsep dan penggunaan dari multi thread

B. Dasar Teori

Pemrograman multithread merupakan konsep penting dalam networking menggunakan Java, misalkan untuk melakukan beberapa tugas yang berbeda dalam satu waktu pada jaringan client-server. Sebelum pembahasan multithread, maka penting untuk memahami perbedaan antar pemrograman single-thread, pemrograman multiprocess, dan pemrograman multi-thread.

Dalam pemrograman single-thread, setiap statemen masing-masing akan dieksekusi satu persatu layaknya sebuah antrian. Pemrograman seperti ini biasa digunakan pada bahasa pemrograman lawas, dimana setiap kode akan dibaca oleh Central Processing Unit (CPU). Pada pemrogaman single-thread ini, apabila suatu statement tidak tereksekusi, maka statement berikutnya juga tidak akan tereksekusi. Dengan kata lain, apabila sebuah variable sedang diakses oleh salah satu statement maka dapat dijamin jika variable itu tidak diakses oleh statement yang lain. Jadi, programmer dapat dengan mudah menentukan statement mana yang mengalami masalah apabila terjadi error pada baris program tersebut.



Gambar 1. Proses single dan multi thread

Berbeda dengan single-thread yang mengeksekusi setiap statement secara berurutan, pada multiproses setiap statement dapat dieksekusi sendiri-sendiri. Sedangkan untuk pemrograman multithread, maka dapat dilihat contohnya pada layar desktop ketika user menjalankan beberapa aplikasi GUI. Berbeda dengan pemrograman multiproses, pada multithread, statement berjalan terpisah namun masih menggunakan

memory yang sama pada komputer, sedangkan pada pemrograman multiproses, memory yang digunakan oleh setiap eksekusi statement akan berbeda.

Multithread pada Java

Eksekusi thread pada java diimplementasikan oleh kelas java.lang.Thread. sedangkan kode untuk tugas-tugas yang dirancang untuk dijalankan di thread direpresentasikan pada kelas interface java.lang.Runnable.

Thread Class

Kelas java.lang.Thread menyediakan method untuk memulai (start), menghentikan (stop), dan melanjutkan (resume) thread, serta untuk mengontrol aspek lain seperti prioritas thread. Diantara method-method yang didefinisikan kelas thread ditunjukkan pada Tabel 1.

Tabel 1. Method pada kelas Thread Java

Method	Deskripsi
static Thread currentThread()	Mendapatkan referensi object Thread yang sedang dieksekusi.
String getName()	Mendapatkan nama dari thread.
int getPriority()	Mendapatkan prioritas dari thread nilainya 1 – 10. Semakin tinggi nilainya maka prioritas thread tsb semakin tinggi.
Boolean isAlive()	Mengetes thread apakah masih aktif!
void join() void join(long millis) void join(long millis, int nanos)	Menunggu hingga thread ini selesai dieksekusi. .
void run()	Method yang pertama kali akan dieksekusi saat thread dibuat.
String setName()	Menset nama dari thread.
static void sleep(long millis) static void sleep(long millis, int nanos)	Menangguhkan eksekusi dari thread yang sedang berjalan untuk sementara waktu
void start()	Byte output Stream yang menambahkan method untuk memudahkan proses menulis ke suatu output. Method yang ditambahkan adalah print() dan println(). Object yang referensinya dipegang oleh **Sytem.out** juga bertipe **PrintStream**.
void setDaemon(Boolean on)	Bila nilai on adalah true maka akan memanggil thread ini terlebih dahulu sebelum menjalankan

Cara mudah untuk menggunakan kelas Thread adalah dengan melakukan extend dan meng-override method run(). Berikut ini contoh yang menunjukkan bagaimana melakukan extend terhadap kelas Thread dan memulai beberapa kejadian yang berkelanjutan, masing-masing pada Thread yang berbeda.

C. Tugas Pendahuluan

Buatlah desain flowchart untuk setiap soal dalam percobaan

D. Percobaan

D.1. Latihan

1. Pemakaian single thread

```
import java.time.LocalTime; // import the LocalTime class

class Proses {
    int data;
    public void setProses (int a) {
        LocalTime waktu = LocalTime.now();
        this.data = a;
        for (int i = 0; i<10; i++) {
            System.out.println("Proses: "+ data + " time: "+ waktu);
        }
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

public class LatSingleThread {
    public static void main(String[] args) {
        Proses proses1 = new Proses();
        Proses proses2 = new Proses();
        proses1.setProses(1);
        proses2.setProses(2);
    }
}
```

Amati hasilnya dan perhatikan bahwa prosesnya dilakukan secara berurutan dari atas ke bawah.

2. Pemakaian multi thread yang sederhana

```
public class Lat1 extends Thread {
  public static void main(String[] args) {
    Lat1 thread = new Lat1();
    thread.start();
    System.out.println("This code is outside of the thread");
  }
  public void run() {
    System.out.println("This code is running in a thread");
  }
}
```

3. Pemakaian multi thread dengan berbeda class

```
import java.time.LocalTime; // import the LocalTime class

class Runner1 extends Thread{
   @Override
    public void run() {
       LocalTime waktu = LocalTime.now();
       for (int i = 0;i <=10;i++){
           System.out.println("Runner1 : "+i+" time:"+waktu);
           try {
               Thread.sleep(1000);
           } catch (InterruptedException e) {
               e.printStackTrace();
           }
       }
    }
}

class Runner2 extends Thread{
    @Override
    public void run() {
        LocalTime waktu = LocalTime.now();
        for (int i = 0;i <=10;i++){
            System.out.println("Runner2 : "+i+" time: "+waktu);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
public class LatThread2 {
    public static void main(String[] args) {
        Runner1 t1 = new Runner1();
        Runner2 t2 = new Runner2();
        t1.start();
        t2.start();
        try { //main thread menunggu kedua thread ini finish
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.print("finish----------------");
    }
}
```

Amati hasilnya, dan bandingkan hasilnya dengan program no 1.

4. Pemakaian multi thread pada satu kelas yang sama

```
import java.util.logging.Level;
import java.util.logging.Logger;
import java.time.LocalTime; // import the LocalTime class

class HaloThread extends Thread {
    String s;
    public HaloThread (String ss){
        this.s = ss;
    }
    @Override
    public void run(){
        LocalTime waktu = LocalTime.now();
        for (int i = 0; i<10; i++) {
            System.out.println(s+" "+i + " : Time " + waktu );
            try {
                sleep(3000);}
            catch (InterruptedException ex) {
Logger.getLogger(HaloThread.class.getName()).log(Level.SEVERE,null,ex);
            }
        }
    }
}

public class LatThread1 {
    public static void main(String[] args) {
        // TODO code application logic here
        HaloThread h1 = new HaloThread(" TH 1 ");
        HaloThread h2 = new HaloThread(" TH 2 ");
        HaloThread h3 = new HaloThread(" TH 3 ");
        h1.start();
        h2.start();
        h3.start();
    }
}
```

D.2. Permasalahan

1. Buatlah program:
    a. Buatlah sebuah kelas **cetakGanjil** yang merupakan turunan dari class thread dimana dalam kelas ini terdapat prosedur yang pertama kali dijalankan. Output dari proses adalah bilangan ganjil dari angka 1..10 dan ditampilkan di layar.
    b. Buatlah sebuah kelas **cetakGenap** yang merupakan turunan dari class thread dimana dalam kelas ini terdapat prosedur yang pertama kali dijalankan. Output dari proses adalah bilangan genap dari angka 1..10 dan ditampilkan di layar.
    c. Buat kelas utama dengan nama **ThreadUtama** yang didalamnya menjalankan kelas **cetakGanjil** dan **cetakGenap** secara bersama-sama.

2. Buatlah class seperti pada nomor 1 hanya saja class **cetakGanjil** dan **cetakGenap** mengimplementasikan interface **Runnable**.
3. Buatlah program yang mampu menjumlahkan 3 buah array bertipe integer berikut:

   Array 1: 6 9 1 2 3
   Array 2: 7 11 6 4 3
   Array 3: 5 4 3 2 1
   ---------- +
   Hasil = 18 24 10 8 7

   Dengan aturan terdapat 5 thread dimana thread 1-5 tugasnya menjumlahkan array sesuai dengan indexnya, misal:
   Thread1: 6+7+5 = 18
   Thread2: 9+11+4=24
   dst
   Total: 67

E. Laporan Resmi :

1. Analisalah semua program diatas dan buat kesimpulan.

Referensi:

1. Override: https://www.duniailkom.com/tutorial-oop-java-pengertian-method-overriding/
2. Array: https://www.w3schools.com/java/java_arrays.asp