MODUL 5
“MULTITHREAD”

Oleh

SAFIRA MAYA SHOVIE, S.Pd

“Multithread”

A. TUJUAN PEMBELAJARAN

1. Memahami konsep dasar instruksi *thread* dalam pemrograman aplikasi berorientasi obyek.
2. Merancang, membuat, dan menguji program aplikasi berorientasi obyek dengan penerapan instruksi *thread* dan *multithread*.

B. DASAR TEORI

Saat ini komputer bukan hanya dituntut untuk dapat melakukan banyak pekerjaan dalam waktu yang cepat. Tapi juga dituntut untuk dapat melakukan beberapa pekerjaan sekaligus dalam satu waktu. Kita sering melakukan aktifitas browsing internet atau office work pada komputer.Disaat bersamaan kita juga mendengarkan music dengan media player di komputer, melakukan proses printing, melakukan download, dan pekerjaan lainnya. Pekerjaan tersebut dapat dilakukan secara bersamaan karena komputer memiliki kemampuan multitasking.

Multitasking adalah proses mengeksekusi beberapa tugas secara simultan (bersamaan). Multitasking dapat dilakukan dengan dua cara:

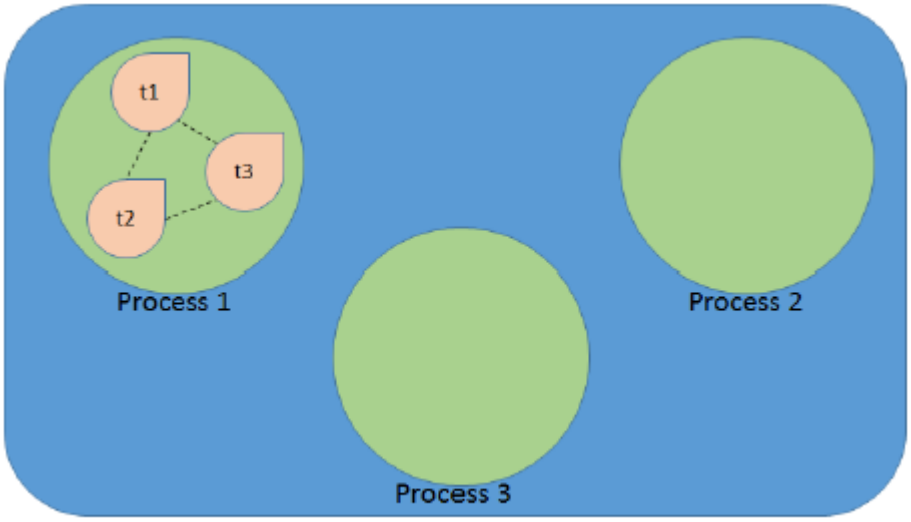
- Proses-based Multitasking (Multiprocessing)
- Thread-based Multitasking (Multithreading)

Multiprocessing adalah menjalankan beberapa proses dalam waktu bersamaan. Yang di maksud proses disini adalah heavyweight process. Setiap proses memiliki alamat sendiri di memori (Setiap proses mengalokasikan area memori terpisah). Biaya komunikasi antar proses tinggi. Perpindahan dari satu proses ke proses yang lain membutuhkan beberapa waktu untuk saving dan loading register, pemetaan memori, update list, dan proses lain.

Multithreading menjalankan beberapa thread dalam waktu bersamaan. Beberapa thread berbagi alamat memori yang sama. Thread merupakan sub-proses yang ringan (lightweight).Biaya komunikasi antar thread rendah. Perpindahan dari satu thread ke thread lain berlangsung cepat.

Multithreading merupakan sebuah konsep untuk dapat menjalankan task atau tugas lebih dari satu secara paralel, sehingga dengan konsep ini task yang banyak akan cepat selesai karena tidak saling tunggu untuk menyelesaikan task. Selain itu dengan menggunakan konsep multithreading Anda benar-benar akan memanfaatkan semua core yang ada dalam processor komputer/laptop Anda.

Thread berjalan didalam sebuah proses. Dalam sebuah OS dapat berjalan beberapa proses sekaligus. Dalam Sebuah proses dapat berjalan beberapa thread. Gambar berikut adalah ilustrasinya.



Thread memiliki 5 state dalam thread life cycle (new, runnable, running, non-runnable,terminated). Namun menurut dokumentasi Sun, hanya ada 4 state dalam thread life cycle dalam java (new, runnable, non-runnable, terminated). Tetapi untuk lebih memahami thread, dalam modul ini dijelaskan dengan 5 state.

1. New

   Kondisi ketika kita telah membuat instance dari class Thread namun belum memanggil method start()

2. Runnable

   Kondisi ketika method start() telah dipanggil, tetapi thread scheduler belum memilih thread tersebut untuk menjadi thread berjalan

3. Running

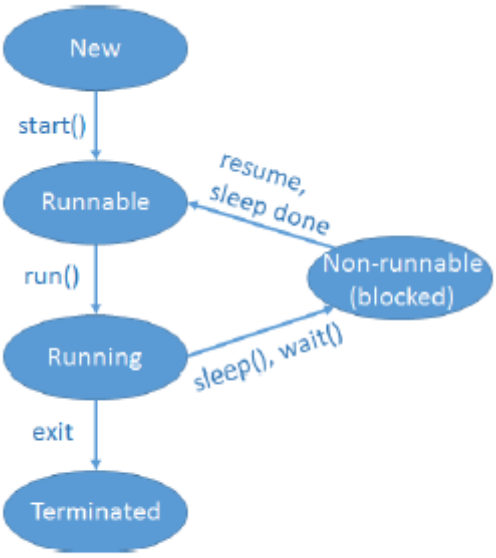
   Kondisi ketika thread telah di start dan thread scheduler telah memilih thread tersebut untuk berjalan

4. Non-Runnable (blocked)

   Kondisi ketika thread masih aktif, namun tidak memenuhi syarat untuk running. Contohnya ketika method sleep() sedang dipanggil

5. Terminated (dead)

   Kondisi ketika thread berhenti berjalan. Yaitu ketika keluar dari run() method



Thread Life Cycle

Java menyediakan class Thread dan interface Runnable untuk melakukan Multithreading. Untuk membuat sebuah thread kita dapat meng-extends class Thread atau dengan implements interface Runnable. Kita juga bisa membuat thread dengan langsung membuat instance dari class Thread.

Dengan menggunakan thread, aplikasi tidak akan memblokir pengguna (doesn't block the user) karena thread bersifat independen dan kita dapat melakukan banyak proses dalam secara bersamaan. Kita dapat melakukan banyak sub-proses secara bersamaan sehingga mempercepat waktu proses. Threads bersifat independent, exception yang terjadi dalam sebuah thread tidak mempengaruhi thread lain.

Thread dapat diimplementasikan untuk proses-proses seperti proses download/upload data ke server, proses read/write data dari file yang membutuhkan waktu lama, proses looping yang membutuhkan waktu lama, proses training/learning data, serta proses komputasi lain yang membutuhkan waktu cukup lama.

C. LATIHAN

Latihan 1

```
package LatThread;

public class tryClass {
   public static void main (String [] args) {
     thread1 T1 = new thread1();
     thread2 T2 = new thread2();
     T1.start();
     T2.start();
   }
}
class thread1 extends Thread {
  @Override
  public void run() {
    while (true) {
      System.out.println("thread 1 running");
      try {
        Thread.sleep(1000);
      } catch (InterruptedException IE) {
        System.out.println("thread interupted");
      }
    }
  }
}
class thread2 extends Thread {
  @Override
  public void run() {
    for (int i = 0; i < 100; i++) {
      System.out.println("thread 2 running "+ i);
      try {
        Thread.sleep(500);
      } catch (InterruptedException IE) {
        System.out.println("thread interupted");
      }
    }
    System.out.println("thread 2 complete");
  }
}
```

Latihan 2

```
package LatThread;

public class tryClass {
  public static void main (String [] args) {
    thread1 th1 = new thread1();
    thread2 th2 = new thread2();
    Thread T1 = new Thread(th1);
    Thread T2 = new Thread(th2);
    T1.start();
    T2.start();
  }
}

class thread1 implements Runnable {
  @Override
  public void run() {
    while (true) {
      System.out.println("thread 1 running");
      try {
        Thread.sleep(1000);
      } catch (InterruptedException IE) {
        System.out.println("thread interupted");
      }
    }
  }
}
class thread2 implements Runnable {
  @Override
  public void run() {
    for (int i = 0; i < 100; i++) {
      System.out.println("thread 2 running "+ i);
      try {
        Thread.sleep(500);
      } catch (InterruptedException IE) {
        System.out.println("thread interupted");
      }
    }
    System.out.println("thread 2 complete");
  }
}
```

Latihan 3

```
package LatThread;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ReadWrite1 implements Runnable {
```

```
//
public static int countwriter = 0;
public static int countreader = 0;
//
public static final int READER = 0;
public static final int WRITER = 1;
//
private int type;
private int id;
private String strid;

public ReadWrite1(int type) {
    this.type = type;
    this.strid = (type == READER) ? ("Reader " + (this.id = ++countreader)) : ("Writer " + (this.id = ++countwriter));
}
```

```
public static void main(String[] argv) {
    (new Thread(new ReadWrite1(READER))).start();
    (new Thread(new ReadWrite1(READER))).start();
    (new Thread(new ReadWrite1(WRITER))).start();
}

public void run() {
    if (type == READER) {
        read();
    } else if (type == WRITER) {
        write();
    }
}

//************************************************************
// Reader
//************************************************************
/**
 *
 */

public void read() {
    System.out.println(strid + ": start!");
    while (true) {
        System.out.println(strid + ": reading!");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ex) {
            Logger.getLogger(ReadWrite1.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
//************************************************************
//************************************************************
// Writer
//************************************************************
```

```
public void write() {
    System.out.println(strid + ": start!");
    while (true) {
        System.out.println(strid + ": writing!");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ex) {
            Logger.getLogger(ReadWrite1.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
//************************************************************
```

Latihan 4

```
package coba.thread;
public class CobaThreadRunnable {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Thread buku1 = new Thread(new Buku("Buku-1"));
        Thread buku2 = new Thread(new Buku("Buku-2"));
        buku1.start();
        buku2.start();
    }
}

class Buku implements Runnable {

    String nama;
// konstruktor
    public Buku(String id) {
        nama = id;
    }


   public void run() {
       for (int i = 1; i < 9; i++) {
           try {
               Thread.currentThread().sleep(1000);
           } catch (InterruptedException ie) {
               System.out.println("Terinterupsi");
           }
           System.out.println("Thread " + nama + ":Posisi" + i);
       }
   }
}
```

Latihan 5

```
package coba.thread;
public class CobaThread {
    public static void main (String [] args) {
    Topi topi1 = new Topi("Topi-1");
    Topi topi2 = new Topi("Topi-2");
    topi1.start();
    topi2.start();
    }
}
    class Topi extends Thread {
        // konstruktor
        public Topi (String id) {
            super (id);
        }
        // Mendefinisikan sendiri run()
    @Override
    public void run() {
        String nama = getName();
        for (int i=1; i<7; i++) {
            try {
                sleep(1000); // Tunggu 1 detik
}
            catch(InterruptedException ie) {
                System.out.println("Terinterupsi");
            }
            System.out.println("Thread " + nama + ":Posisi" + i );
        }
```

```
    }
}
```

Latihan 6

Silahkan membuat program multithread dengan mengacu contoh latihan program diatas serta berikan penjelasan!

Catatan :

1. Silahkan mengerjakan semua latihan yang terdapat diatas, bisa didiskusikan dengan teman kalian. Tetapi untuk pengerjaan tugasnya tetap individu.
2. Materi diatas hanya sedikit rangkuman dari berbagai fungsi kode program yang sudah saya buat. Silahkan untuk pemantaban pemahaman lebih dalam, kalian cari referensi contoh program yang lain dengan menggunakan beberapa fungsi dari kode program diatas.
3. Deadline pengumpulan tugas, 3 hari setelah materi dibagikan. Jika telat dalam pengumpulan, tugas tidak diterima, terkecuali bila ada kendala, misal : laptop rusak dsb, maka setelah laptop kembali normal tugas bisa menyusul.
4. Tugas dikumpulkan melalui link berikut: https://drive.google.com/drive/folders/1fY2O03GaYMWldM8ijiNvdtlyOyvfqM21?usp=sharing, dengan catatan yang dikumpulkan adalah project pada netbeans yang sudah dikerjakan, dan laporan yang berisi screenshoot syntax program dan output program serta penjelasan program.
5. Jangan lupa format folder tugas dan laporan yaitu **No.Absen_Nama Lengkap_Kelas.**

SELAMAT MENGERJAKAN ^_^