Materi IX

Nama : Roy Agus Martin Marbun

Kelas : XII – RPL

Mata Pelajaran : Pemrograman Berorientasi Objek

BAB II

Operasi *File* dan *Input Output* (IO)

A. Prioritas *Thread*

Untuk menentukn *thread* mana yang akan menerima *control* dari CPU dan akan dieksekusi pertama kali, setiap *thread* akan diberikan sebuah prioritas. Sebuah prioritas adalah sebuah nilai integer dari angka 1 sampai dengan 10, dimana semakin tinggi prioritas dari sebuah *thread*, berarti semakin besar kesempatan dari *thread* tersebut untuk dieksekusi terlebih dahulu.

Sebagai contoh, asumsikan bahwa ada dua buah *thread* yang berjalan bersama-sama. *Thread* pertama akan diberikan prioritas nomor 5, sedangkan *thread* yang kedua memiliki prioritas 10. Anggplan bahwa *thread* pertama telah berjalan pada saat *thread* kedua dijalankan. *Thread* kedua akan menerima *control* dari CPU dan akan dieksekusi pada saat *thread* kedua tersebut memiliki priorits yang lebih tinggi dibandingkan *thread* yang pada saat itu tengah berjalan. Salah satu contoh dari skenario ini adalah *context switch*.

Sebuah *context switch* terjadi apabila sebagian dari *thread* telah dikontrol oleh CPU dari *thread* yang lain. Ada beberapa skenario mengenai bagaimana cara kerja dari *context switch*. Salah satu skenario adalah sebuah *thread* yang sedang berjalan memberikan kesempatan kepada CPU untuk mengontrol *thread* lain sehingga ia dapat berjalan. Dalam kasus ini, prioritas tertinggi dari *thread* adalah *thread* yang siap untuk menerima kontrol dari CPU. Cara yang lain dari *context switch* adalah pada saat sebuah *thread* yang sedang berjalan diambil alih oleh *thread* yang memiliki prioritas tertinggi seperti yang telah dicontohkan sebelumnya.

Hal ini juga mungkin dilakukan apabila lebih dari satu CPU tersedia, sehingga lebih dari satu prioritas *thread* yang siap untuk dijalankan. Untuk menentukan diantara dua *thread* yang memiliki prioritas sama untuk menerima kontrol dari CPU, sangat bergantung kepada sistem operasi yang digunakan. Windows 95/98/NT menggunakan *time-slicing* dan *round-robin* untuk menangani kasus ini. Setiap *thread* dengan prioritas yang sama akan diberikan sebuah jangka waktu tertentu untuk dieksekusi sebelum CPU mengontrol *thread* lain yang memiliki prioritas yang sama. Sedangkan Solaris, ia akan membiarkan sebuah *thread* untuk dieksekusi sampai ia menyelesaikan tugasnya atau sampai ia secara suka rela membiarkan CPU untuk mengontrol *thread* yang lain.

B. Class Thread

1. Constructor

   Thread memiliki delapan constructor. Berikut adalah beberapa contructor tersebut.

Thread Contructors	Kegunaan
Thread()	Membuat object Thread yang baru
Thread(String name)	Membuat sebuah object thread dengan memberikan penamaan yang spesifik
Thread(Runnable target)	Membuat sebuah object Thread yang baru berdasar pada object Runnable. Target menyatakan sebuah object dimana method run dipanggil
Thread(Runnable targe, String name)	Membuat sebuah object Thread yang baru dengan nama yang spesifik dan berdasarkan pada object Runnable

2. Constants

   Class Thread juga menyediakan beberapa constants sebagai nilai prioritas. Tabel berikut ini adalah rangkuman dari class Thread.

Thread Constants	Kegunaan
public final static int MAX_PRIORITY	Nilai prioritas maksimum, 10
public final static int MIN_PRIORITY	Nilai prioritas minimum, 1
public final static int NORM_PRIORITY	Nilai default prioritas, 5

3. Methods

   Method-method inilah yang disediakan dalam class Thread.

Thread Methods	Kegunaan
public static Thread currentThread()	Mengembalikan sebuah reference kepada thread yang sedang berjalan
public final String getName()	Mengembalikan nama dari thread
public final void setName(String name)	Mengulang pemberian nama thread sesuai denagn argument name. Hal ini dapat menyebabkan SecurityException
public final int getPriority()	Mengembalikan nilai prioritas yang telah diberikan kepada thread tersebut.
public final Boolean isAlive()	Menunjukkan bahwa thread tersebut sedang berjalan atau tidak
public final void join([long millis, [int nanos]])	Sebuah overloading method. Sebuah thread yang sedang berjalan, harus menunggu sampai thread tersebut selesai (jika tidak ada parameter-parameter spesifik), atau sampai waktu yang telah ditentukan habis
public static void sleep(long millis)	Menunda thread dalam jangka waktu milis. Hal ini dapat menyebabkan InterruptedException
public void run()	Eksekusi Thread dimulai dari method ini
public void start()	Menyebabkan eksekusi dari thread berlangsung dengan cara memanggil method run

C. Sebuah contoh Thread

Contoh dari thread pertama pada sebuah counter yang sederhana.

```
Import javax.swing.*;
Import java.awt.*;

Class CountDownGUI extends JFrame{
        JLabel label;
        CountDownGUI(String title){
                super(title);
                label= new JLabel("Start count !");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                getContentPane().add(new Panel(), BorderLayout.WEST);
                getContentPane().add(label);
                getSize(300, 300);
                setVisible(true);
        }
        void startCount(){
                try{
                        for(int i = 10; i > 0; i--){
                                Thread.sleep(1000);
                                label.setText(i + "");
                        }
                        Thread.sleep(1000);
                        label.setText("Count down complete.");
                        Thread.sleep(1000);
                } catch(InterruptedException ie){
                }
                label.setText(Thread.currentThread().toString());
        }
        public static void main (String args[]){
                CountDownGUI cdg = new CountDownGUI("Count down GUI");
                cdg.startCount();
        }
}
```