Bab 1

Class Timer

1.1 Tujuan

Setelah mempelajari bab ini, siswa di harapkan :

- Memahami dan menggunakan class Timer dan TimerTask
- Mengolah waktu dengan Timer dan TimerTask
- Menentukan waktu yang sesuai untuk penjadwalan
- Dapat membuat animasi dari Timer dan TimerTask

1.2 Pengantar

Class Timer merupakan salah satu pengembangan dari versi Java 1.3, dalam lingkup J2SE berupa pembentukan class-class yang mempermudah pengaturan penjadwalan proses agar dapat dieksekusi melalui pengaksesan fungsi dasar thread. Pembaharuan diatas juga dapat dimanfaatkan oleh pengembang pemrograman berbasis mobile (J2ME) karena class baru diatas juga telah menjadi bagian dalam MIDP (*Mobile Information Device Profile*).

1.3 Memanfaatkan class Timer

Proses didefinisikan dan dijadwalkan menggunakan dua class yaitu :

- `java.util.TimerTask`
- `java.util.Timer`

Class `TimerTask` merupakan class abstract yang berfungsi sebagai class dasar untuk semua penjadwalan proses. Sedangkan class `Timer` membuat dan mengatur thread dimana proses dieksekusi.

Untuk mendefinisikan sebuah proses, buatlah sebuah subclass dari `TimerTask` yang meng-*implements* method run, sebagai contoh :

```
import java.util. TimerTask;

public class MyTask extends TimerTask {
    public void run() {
        System.out.println("Menjalankan Task");
    }
}
```

Jika method run tergolong cukup dikenal, hal ini dikarenakan `TimerTask` meng-*implements* interface `java.lang.Runnable`. Class `Timer` menjalankan method run untuk menjalankan proses. Method seharusnya menampilkan proses dan keluar secepatnya karena hanya satu proses untuk setiap object `Timer` dapat dieksekusi kapanpun.

Setelah mendefinisikan sebuah proses, anda dapat menjadwalkannya dengan cara menciptakan sebuah instance dari `Timer` dan menjalankan method penjadwalan, seperti dapat dilihat sebagai berikut :

```
import java.util.TimerTask;
import java.util.Timer;

Timer     timer = new Timer();
TimerTask task = new MyTask();

// tunggu 10 detik untuk menjalankan aplikasi
timer.schedule( task, 10000 );

// tunggu 5 detik sebelum di eksekusi, maka
// mengeksekudi setiap 5 detik
timer.schedule( task, 5000, 10000 );
```

terdapat empat versi method penjadwalan, setiap penjadwalan proses dilakukan pada suatu waktu tertentu (khususnya menggunakan sebuah object `Date`) atau setelah suatu *delay*(tunda waktu) tertentu (dalam *miliseconds*). Anda dapat menjadwalkan proses untuk dijalankan sekali atau diulang terus menerus pada jangka waktu tertentu. Terdapat juga suatu method `scheduleAtFixedRate` yang menjadwalkan proses untuk dieksekusi secara berulang dalam interval relative terhadap waktu eksekusi yang telah dijadwalkan terhadap eksekusi pertama. Jika proses eksekusi ditunda (untuk kasus garbage collection), dua atau lebih sub urutan eksekusi dijadwalkan dalam interval yang lebih pendek untuk “ditangkap”

Setiap object Timer menciptakan dan mengatur sebuah fungsi thread tunggal. Sebuah timer tunggal biasanya mewakili kebutuhan semua aplikasi tunggal, akan tetapi disini tidak ada batasan berapa-pun jumlah timer yang dapat Anda dibuat. Anda dapat menghentikan sebuah timer kapanpun dan melakukan terminasi bagian threadnya dengan memanggil method cancel timer. Dengan catatan jika timer dihentikan sekali, timer tersebut tidak dapat di *restart* – Anda harus membuat sebuah object Timer dan menjadwalkan kembali proses yang ingin Anda eksekusi. Object Timer menyimpan fungsi thread, disini tidak perlu menampilkan beberapa sinkronisasi nyata jika Anda memanggil sebuah object pada thread yang berbeda.

Setelah Anda menjadwalkan sebuah proses, Anda dapat menghentikan eksekusinya dengan memanggil method cancelnya. Hal ini sering terjadi dalam method run dari sebuah proses. Panggil method cancel bersama dengan method run menjamin bahwa eksekusi dari proses tersebut adalah proses yang paling akhir. Dalam hal ini juga memungkinkan method dapat dipanggil pada point manapun, asalkan sebelumnya, proses dijadwalkan untuk dieksekusi pertama kali.

Berikut ini adalah contoh sederhana dari suatu MIDlet yang menggunakan timer untuk menampilkan splash screen sederhana dari sebuah gambar ke sebuah teks.

```
import java.util.Timer;
import java.util.TimerTask;
...

public class MidletSplashTimer extends MIDlet {
    ...
    private Timer timer;
    private TimerTask timerTask;

    protected void startApp() {
        ...
        display.setCurrent(ttCanvas);
    }

    void stopTimer() {
        timer.cancel();
        timerTask.cancel();
    }
```

```
    void startTimer() {
        timer = new Timer();
        timerTask = new TimerTask() {
           public void run() {
              display.setCurrent(new SuccesCanvas());
           }
        };
        timer.schedule(timerTask, 3000);
    }
};
```

```
import javax.microedition.lcdui.*;

public class SplashTimerCanvas extends Canvas {
    private MidletSplashTimer midletTT;
    ...

    public SplashTimerCanvas(MidletSplashTimer midlet) {
        ...
    }

    protected void paint(Graphics g) {
        ...
        // splash screen
    }

    class SuccesCanvas extends Canvas {
        protected void paint(Graphics g) {
        ...
    }
}
```

MIDlet MidletSplashTimer menggunakan sebuah object Timer, timer, untuk menjadwalkan eksekusi dari subclass TimerTask, FieldMover, setiap 3000 milidetik. paint pada class SplashTimerCanvas menggambarkan splash screen yang akan ditampilkandan paint pada class SuccesCanvas menggambarkan teks yang akan ditampilkan sesudah splash screen ditampilkan.