MODUL PRAKTIKUM

Thread

Versi 1.3

1. Tujuan

- Mendefiniskan threads
- Mengerti perbedaan state dalam threads
- Mengerti konsep prioritas dalam threads
- Mengetahui bagaimana menggunakan method didalam class Thread
- Membuat sendiri sebuah thread
- Menggunakan sinkronisasi pada thread yang bekerja bersama-sama dan saling bergantung satu dengan yang lainya
- Memungkinkan thread untuk dapat berkomunikasi dengan thread lain yang sedang berjalan
- Mengerti dan menggunakan kemampuan concurency

2. Latar Belakang

Pada bab-bab sebelumnya Anda terbiasa untuk membuat program yang berururutan/sekuensial. Sebuah program sekuensial berarti sebuah program yang hanya memiliki satu aliran eksekusi. Setiap eksekusi, ia memiliki sebuah titik awal eksekusi, kemudian sebuah sekuen eksekusi, dan kemudian berakhir. Selama runtime, pasti hanya satu proses yang telah dieksekusi. Bagaimanapun juga, di dunia nyata, pasti dibutuhkan sesuatu yang dapat mengatur proses yang terjadi dan berjalan bersama-sama.Oleh karena itu, thread hadir untuk menjadi solusi dalam mengatasi permasalahan tersebut.Sebuah thread merupakan sebuah pengontrol aliran program. Untuk lebih mudahnya, bayangkanlah thread sebagai sebuah proses yang akan dieksekusi didalam sebuah program tertentu. Penggunaan sistem operasi modern saat ini telah mendukung kemampuan untuk menjalankan beberapa program. Misalnya, pada saat Anda mengetik sebuah dokumen di komputer Anda dengan menggunakan text editor, dalam waktu yang bersamaan Anda juga dapat mendengarkan musik, dan surfing lewat internet di PC Anda. Sistem operasi yang telah terinstal dalam computer Anda itulah yang memperbolehkan Anda untuk menjalankan multitaskting. Seperti itu juga sebuah program (ibaratkan di PC Anda), ia juga dapat mengeksekusi beberapa proses secara bersama-sama(ibaratkan beberapa aplikasi berbeda yang bekerja pada PC Anda). Sebuah contoh aplikasi adalah HotJava browser yang memperbolehkan Anda untuk browsing terhadap suatu page, bersamaan dengan mendownload object yang lain, misalnya gambar, memainkan animasi, dan juga file audio pada saat yang bersamaan.

3. Percobaan

Percobaan 1 Counter Down Demo :

```
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);
      setSize(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();
   }
}
```

Percobaan 2 Menulis Object Threads sebanyak 100 x :

```
class PrintNameThread extends Thread {
   PrintNameThread(String name) {
      super(name);
      // menjalankan thread dengan satu kali instantiate
      start();      }
   public void run() {
      String name = getName();
      for (int i = 0; i < 100; i++) {
         System.out.print(name);
      }
   }
}

class TestThread {
   public static void main(String args[]) {
      PrintNameThread pnt1 = new PrintNameThread("A");
      PrintNameThread pnt2 = new PrintNameThread("B");
      PrintNameThread pnt3 = new PrintNameThread("C");
      PrintNameThread pnt4 = new PrintNameThread("D");
   }
}
```

Percobaan 3 Implementasi interface Runnable :

```
class PrintNameThread implements Runnable {
   Thread thread;
   PrintNameThread(String name) {
      thread = new Thread(this, name);
      thread.start();
   }
   public void run() {
      String name = thread.getName();
      for (int i = 0; i < 100; i++) {
         System.out.print(name);
      }
   }
}

class TestThread {
   public static void main(String args[]) {
      new PrintNameThread("A");
      new PrintNameThread("B");
      new PrintNameThread("C");
      new PrintNameThread("D");
   }
}
```

Percobaan 4 Contoh penggunaan method Join:

```
class PrintNameThread implements Runnable {
   Thread thread;
   PrintNameThread(String name) {
      thread = new Thread(this, name);
      thread.start();
   }
   public void run() {
      String name = thread.getName();
      for (int i = 0; i < 100; i++) {
         System.out.print(name);
      }
   }
}

class TestThread {
   public static void main(String args[]) {
      PrintNameThread pnt1 = new PrintNameThread("A");
      PrintNameThread pnt2 = new PrintNameThread("B");
      PrintNameThread pnt3 = new PrintNameThread("C");
      PrintNameThread pnt4 = new PrintNameThread("D");
      System.out.println("Running threads...");
      try {
         pnt1.thread.join();
         pnt2.thread.join();
         pnt3.thread.join();
         pnt4.thread.join();
      } catch (InterruptedException ie) {
      }
      System.out.println("Threads killed."); //dicetak terakhir
   }
}
```

Percobaan 5 Mencetak String tanpa sinkronisasi:

```
class TwoStrings {
   static void print(String str1, String str2) {
      System.out.print(str1);
      try {
         Thread.sleep(500);
      } catch (InterruptedException ie) {
      }
      System.out.println(str2);
   }
}

class PrintStringsThread implements Runnable {
   Thread thread;
   String str1, str2;
   PrintStringsThread(String str1, String str2) {
      this.str1 = str1;
      this.str2 = str2;
      thread = new Thread(this);
      thread.start();
   }
   public void run() {
      TwoStrings.print(str1, str2);
   }
}
class TestThread {
   public static void main(String args[]) {
      new PrintStringsThread("Hello ", "there.");
      new PrintStringsThread("How are ", "you?");
      new PrintStringsThread("Thank you ", "very much!");
   }
}
```

Percobaan 6 Sinkronisasi pertama:

```
class TwoStrings {
   synchronized static void print(String str1, String str2) {
      System.out.print(str1);
      try {
         Thread.sleep(500);
      } catch (InterruptedException ie) {
      }
      System.out.println(str2);
   }
}

class PrintStringsThread implements Runnable {
   Thread thread;
   String str1, str2;
   PrintStringsThread(String str1, String str2) {
      this.str1 = str1;
      this.str2 = str2;
      thread = new Thread(this);
      thread.start();
   }
   public void run() {
      TwoStrings.print(str1, str2);
   }
}

class TestThread {
   public static void main(String args[]) {
      new PrintStringsThread("Hello ", "there.");
      new PrintStringsThread("How are ", "you?");
      new PrintStringsThread("Thank you ", "very much!");
   }
}
```

Percobaan 7 Sinkronisasi kedua:

```
class TwoStrings {
   static void print(String str1, String str2) {
      System.out.print(str1);
      try {
         Thread.sleep(500);
      } catch (InterruptedException ie) {
      }
      System.out.println(str2);
   }
}

class PrintStringsThread implements Runnable {
   Thread thread;
   String str1, str2;
   TwoStrings ts;
   PrintStringsThread(String str1, String str2, TwoStrings ts)
   {
      this.str1 = str1;
      this.str2 = str2;
      this.ts = ts;
      thread = new Thread(this);
      thread.start();
   }
   public void run() {
      synchronized (ts) {
         ts.print(str1, str2);
      }
   }
}

class TestThread {
   public static void main(String args[]) {
      TwoStrings ts = new TwoStrings();
      new PrintStringsThread("Hello ", "there.", ts);
      new PrintStringsThread("How are ", "you?", ts);
      new PrintStringsThread("Thank you ", "very much!", ts);
   }
}
```

Percobaan 8 Produser-Consumer Test:

```
class SharedData {
   int data;
   synchronized void set(int value) {
      System.out.println("Generate " + value);
      data = value;
   }
   synchronized int get() {
      System.out.println("Get " + data);
      return data;
   }
}

class Producer implements Runnable {
   SharedData sd;
   Producer(SharedData sd) {
      this.sd = sd;
      new Thread(this, "Producer").start();
   }
   public void run() {
      for (int i = 0; i < 10; i++) {
         sd.set((int)(Math.random()*100));
      }
   }
}

class Consumer implements Runnable {
   SharedData sd;
   Consumer(SharedData sd) {
      this.sd = sd;
      new Thread(this, "Consumer").start();
   }
    public void run() {
       for (int i = 0; i < 10 ; i++) {
          sd.get();
       }
    }
}

 class TestProducerConsumer {
    public static void main(String args[]) throws Exception {
       SharedData sd = new SharedData();
       new Producer(sd);
       new Consumer(sd);
    }
}
```

Percobaan 9 Produser-Consumer Test Modifikasi:

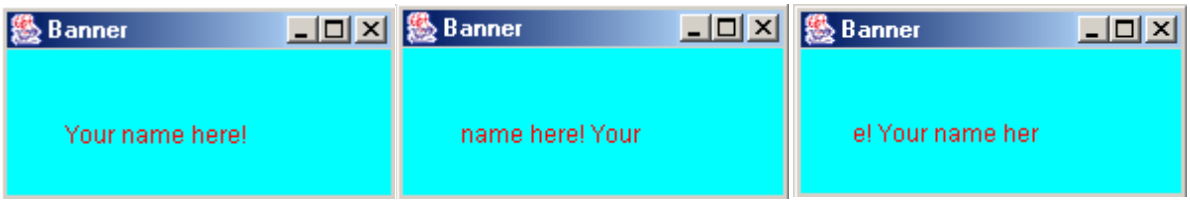
```
class SharedData {
   int data;
   boolean valueSet = false;
   synchronized void set(int value) {
      if (valueSet) {  //baru saja membangkitkan sebuah nilai
         try {
            wait();
         } catch (InterruptedException ie) {
         }
      }
      System.out.println("Generate " + value);
      data = value;
      valueSet = true;
      notify();
   }
   synchronized int get() {
      if (!valueSet) { //produsen belum men-set sebuah nilai
         try {
            wait();
         } catch (InterruptedException ie) {
         }
      }
      System.out.println("Get " + data);
      valueSet = false;
      notify();
      return data;
   }
}

/* Bagian kode ini tidak ada yang berubah*/
class Producer implements Runnable {
   SharedData sd;
   Producer(SharedData sd) {
      this.sd = sd;
      new Thread(this, "Producer").start();
   }
   public void run() {
      for (int i = 0; i < 10; i++) {
         sd.set((int)(Math.random()*100));
      }
   }
}

class Consumer implements Runnable {
   SharedData sd;
   Consumer(SharedData sd) {
      this.sd = sd;
      new Thread(this, "Consumer").start();
   }
```

4. Latihan

4.1 Banner

Dengan menggunakan AWT atau Swing, buatlah sebuah banner sederhana yang akan mencetak string yang dituliskan oleh user. String ini akan ditampilkan secara terus menerus dan program Anda harus memberikan ilustrasi bahwa string tersebut bergerak dari kiri ke kanan. Untuk memastikan bahwa proses perpindahannya tidak terlalu cepat, Anda sebaiknya menggunakan method sleep dari class Thread.

Berikut ini adalah sebuah contoh dimana Anda menuliskan "Your name here!".



Gambar 1.6.1: Contoh pergerakan string