Threads

Objektif

- 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

Objektif

- Memungkinkan thread untuk dapat berkomunikasi dengan thread lain yang sedang berjalan

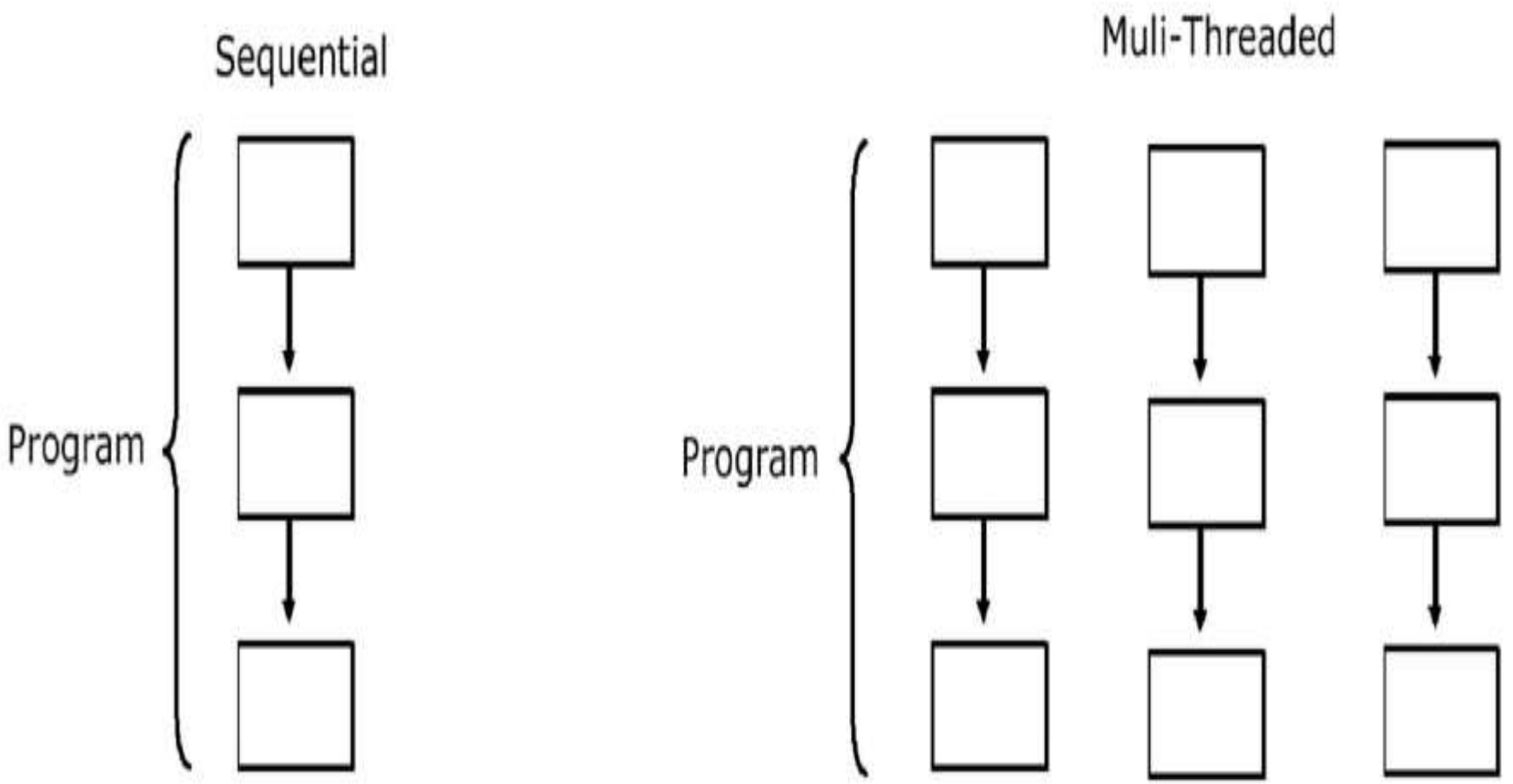
Definisi Thread

- Sebuah thread merupakan sebuah pengontrol aliran program.
- Thread sebagai sebuah proses yang akan dieksekusi didalam sebuahprogram tertentu.
- Penggunaan sistem operasi modern saat ini telah mendukung kemampuan untuk menjalankan beberapa program.
- Sebuah program (ibaratkan di PC Anda), Program juga dapat mengeksekusi beberapa proses secara bersama-sama.

Definisi Thread

- Sebuah contoh aplikasi adalah HotJava browser yang memperbolehkan untuk browsing terhadap suatu page, bersamaan dengan mendownload object yang lain, misalnya gambar, memainkan animasi, dan juga file audio pada saat yang bersamaan.

Definisi Thread



Gambar 1.1: Thread

State dari Thread

Sebuah thread dapat memiliki beberapa state:

- Running

  Sebuah thread yang pada saat ini sedang dieksekusi dan didalam control dari CPU.

- Ready to run

  Thread yang sudah siap untuk dieksekusi, tetapi masih belum ada kesempatan untuk melakukannya.

- Resumed

  Setelah sebelumnya di block atau diberhentikan sementara, state ini kemudian siap untuk dijalankan.

State dari Thread

- Suspended

  Sebuah thread yang berhenti sementara, dan kemudian memperbolehkan CPU untuk menjalankan thread lain bekerja.

- Blocked

  Sebuah thread yang di-block merupakan sebuah thread yang tidak mampu berjalan,

- karena ia akan menunggu sebuah resource tersedia atau sebuah event terjadi.

Prioritas

- Untuk menentukan thread mana yang akan menerima control dari CPU dan akandieksekusi 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.

Constructor

Constructor-constructor Thread
`Thread()`
Membuat sebuah 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 target, String name)`
Membuat sebuah object *Thread* yang baru dengan nama yang spesifik dan berdasarkan pada object *Runnable*.

Constants

- Class *Thread juga menyediakan beberapa constants sebagai nilai prioritas*

Thread Constants
`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.

Method

Method-method Thread
`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 dengan 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.

Method

`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.

Contoh Program tread

```
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();
    }
}
```

Membuat Threads dg (extend) class

- Sebuah thread dapat diciptakan dengan cara menurunkan (extend) class *Thread atau* dengan mengimplementasikan sebuah interface *Runnable*

```
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);
      }
   }
}
```

Membuat Threads dg (extend) class

```
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");
    }
}
```

Membuat Threads dg 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);
        }
    }
}
```

Membuat Threads dg interface *Runnable*

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

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);
        }
    }
}
```

Penggunaan method join

```
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
   }
}
```

Sinkronisasi

- Suatu thread yang membutuhkan resource atau method dari luar sehingga ia membutuhkan komunikasi dengan thread lain.
- Untuk memastikan bahwa hanya satu thread yang mendapatkan hak akses kedalam

  method tertentu, Java memperbolehkan penguncian terhadap sebuah object termasuk

  method-method-nya dengan menggunakan monitor.

Sinkronisasi

- Object tersebut akan menjalankan sebuah monitor implicit pada saat object dari method sinkronisasi dipanggil.
- Sekali object tersebut dimonitor, monitor tersebut akan memastikan bahwa tidak ada thread yang akan mengakses object yang sama.
- Sebagai konsekuensinya, hanya ada satu thread dalam satu waktu yang akan mengeksekusi method dari object tersebut.

```
synchronized (<object>) {
   //statements yang akan disinkronisasikan
}
```

Sebuah contoh yang tidak disinkronisasi

```
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) {
```

Sebuah contoh yang tidak disinkronisasi

```
        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!");
    }
}
```

Contoh sinkronisasi

```
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();
      }
```

Contoh sinkronisasi

```
    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!");
    }
}
```

Method-method untuk komunikasi Interthread
`public final void wait()`
Menyebabkan thread ini menunggu sampai thread yang lain memanggil *notify* atau *notifyAll* method dari object ini. Hal ini dapat menyebabkan *InterruptedException*.
`public final void notify()`
Membangunkan thread yang telah memanggil method *wait* dari object yang sama.
`public final void notifyAll()`
Membangunkan semua thread yang telah memanggil method *wait* dari object yang sama.


Customer
Waiter (waiting)
comes in
notifies the waiter
goes to the customer
chooses order
asks the waiter to wait
waits for the order
gives order
notifies the waiter
takes the order
waits for the order
asks the customer to wait
leaves for the food
eats the food
notifies the customer
returns with the food
leaves the restaurant
waits again

```
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);
    }
}
```

```
Generate 8
Generate 45
Generate 52
Generate 65
Get 65
Generate 23
Get 23
Generate 49

 Get 49
 Generate 35
 Get 35
 Generate 39
 Get 39
 Generate 85
 Get 85
 Get 85
 Get 85
 Generate 35
 Get 35
 Get 35
```

```
Generate 76
Get 76
Generate 25
Get 25
Generate 34
Get 34
Generate 84
Get 84
Generate 48
Get 48
Generate 29
Get 29
Generate 26
Get 26
Generate 86
Get 86
Generate 65
Get 65
Generate 38
Get 38
Generate 46
Get 46
```

```
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;
    }
}
```

```
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);
    }
}
```

Latihan

Banner

Dengan menggunakan AWT atau Swing, buatlah sebuah banner sederhana yang akanmencetak 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!”.


Banner
Your name here!
Banner
name here! Your
Banner
e! Your name her