PRAKTIKUM 5

MULTITHREADING 1

A. TUJUAN PEMBELAJARAN

1. Mampu mengimplementasikan thread dalam sebuah aplikasi
2. Memahami kegunaan thread dalam aplikasi

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:

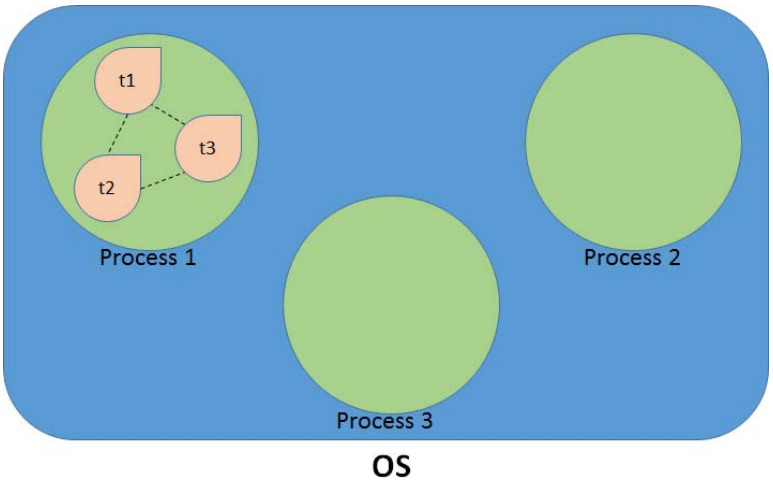
- Proses-based Multitasking (Multiprocessing)
- Thread-based Multitasking (Multithreading)

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.

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

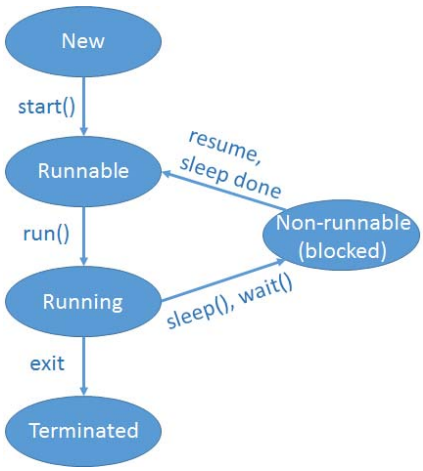
   Kondisi ketika thread telah di start dan thread scheduler telah memilih thread tersebut untuk berjalan

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. TUGAS PENDAHULUAN

1. Apa perbedaan multiprocessing dan multithreading?
2. Apa yang dimaksud dengan thread di dalam java?
3. Jelaskan keuntungan penggunaan thread!
4. Sebutkan contoh proses yang membutuhkan thread!

D. PERCOBAAN

Percobaan 1 : Implementasi thread dengan Thread Class sebagai parent

Salah satu cara untuk membuat thread adalah dengan meng-extend class Thread. Program berikut adalah contoh cara membuat thread dengan meng-extend class thread.

```
3   public class ExtendsThread extends Thread{
4       @Override
5       public void run(){
6           //Tulis proses yang harus dijalankan dengan thread dalam method run()
7           System.out.println("thread is running...");
8       }
9       public static void main(String args[]){
10          ExtendsThread t1 = new ExtendsThread();
11          t1.start();
12      }
13  }
```

Percobaan 2 : Implementasi thread dengan Interface Runnable

Selain dengan meng-extend class thread kita juga dapat membuat thread dengan cara mengimplementasikan interface runnable. Berikut adalah contoh program untuk membuat thread dengan interface runnable.

```
3   public class ImplementsRunnable implements Runnable{
4       @Override
5       public void run(){
6           //Tulis proses yang harus dijalankan dengan thread dalam method run()
7           System.out.println("thread is running...");
8       }
9
10      public static void main(String args[]){
11          ImplementsRunnable m1=new ImplementsRunnable();
12          Thread t1 = new Thread(m1);
13          t1.start();
14      }
15  }
```

Percobaan 3 : Implementasi thread dengan membuat instance dari Class Thread

Selain mengimplementasikan interface runnable atau meng-extend class Thread, kita juga bisa membuat thread dengan cara langsung membuat instance dari class thread.

```
3   public class Example {
4       public static void main(String args[]){
5           Thread t = new Thread(new Runnable() {
6               @Override
7               public void run() {
8                   //Tulis proses yang harus dijalankan dengan thread dalam method run()
9                   System.out.println("thread is running...");
10              }
11          });
12          t.start();
13      }
14  }
```

Percobaan 4 : Start Thread Twice

Apakah sebuah thread dapat dijalankan lebih dari satu kali? Untuk menjawab pertanyaan tersebut silahkan jalankan program berikut ini.

```
1   public class TestThreadTwice1 extends Thread{
2       public void run(){
3           System.out.println("running...");
4       }
5       public static void main(String args[]){
6           TestThreadTwice1 t1=new TestThreadTwice1();
7           t1.start();
8           t1.start();
9       }
10  }
```

Percobaan 5 : Menjalankan Thread dengan memanggil method run()

Dapatkah kita menjalankan thread dengan hanya memanggil method run() saja?

```
class TestCallRun1 extends Thread{
    public void run(){
        System.out.println("running...");
    }
    public static void main(String args[]){
        TestCallRun1 t1=new TestCallRun1();
        t1.run();//fine, but does not start a separate call stack
    }
}
```

Percobaan 6 : Permasalahan ketika memanggil method run()

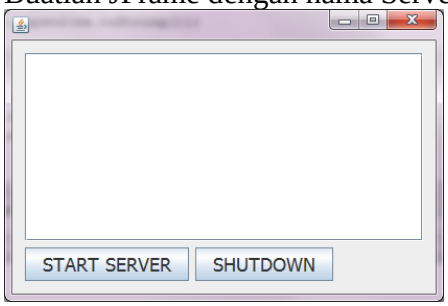
Apa yang terjadi jika kita menjalankan thread dengan memanggil method run? Apa perbedaan memanggil method run() dan start()? Untuk menjawab pertanyaan tersebut silahkan jalankan dua program berikut. Buatlah analisa untuk perbedaan pemanggilan method run() dan start().

```
class TestCallRun2 extends Thread{
    public void run(){
        for(int i=1;i<5;i++){
            try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
            System.out.println(i);
        }
        }
    public static void main(String args[]){
        TestCallRun2 t1=new TestCallRun2();
        TestCallRun2 t2=new TestCallRun2();

        t1.run();
        t2.run();
    }
}

class TestCallStart2 extends Thread{
    public void run(){
        for(int i=1;i<5;i++){
            try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
            System.out.println(i);
        }
        }
    public static void main(String args[]){
        TestCallRun2 t1=new TestCallRun2();
        TestCallRun2 t2=new TestCallRun2();

        t1.start();
        t2.start();
    }
}
```

E. LATIHAN

Latihan 1 : Aplikasi Client-Server tanpa thread

- Buatlah JFrame dengan nama ServerFrame.java seperti berikut.



- Tambahkan variable dan method berikut ke dalam class ServerFrame.java

```
private ServerSocket serverSocket;
private static final int PORT = 6006;

public void startServer() throws IOException {
    try {
        textArea.append("Waiting for client on port " + serverSocket.getLocalPort() + "...\n");
        Socket server = serverSocket.accept();

        textArea.append("Just connected to " + server.getRemoteSocketAddress() + "\n");
        DataInputStream in = new DataInputStream(server.getInputStream());

        textArea.append(in.readUTF() + "\n");
        DataOutputStream out = new DataOutputStream(server.getOutputStream());
        out.writeUTF("Hello, you are connected...");
    } catch (SocketTimeoutException s) {
        System.out.println("Socket timed out!");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

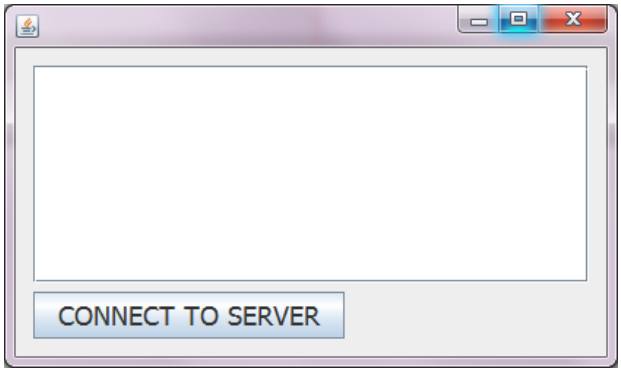
- Tambahkan code berikut dalam button Start

```
try {
    serverSocket = new ServerSocket(PORT);
    startServer();
} catch (IOException ex) {
    textArea.append(ex.toString());
}
```

- Tambahkan code berikut dalam button Shutdown

```
System.exit(0);
```

- Buatlah JFrame dengan nama ClientFrame.java seperti berikut.



- Tambahkan variable dan method berikut pada class ClientFrame.java

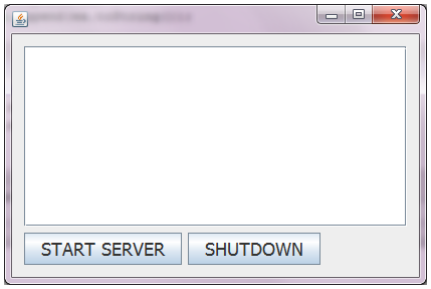
```
private final String serverName = "localhost";
private final int port = 6006;
private Socket client;

public void startClient(){
    while(true){
        try {
            OutputStream outToServer = client.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToServer);
            out.writeUTF("hello i am client");

            InputStream inFromServer = client.getInputStream();
            DataInputStream in = new DataInputStream(inFromServer);
            textArea.append("Server echo : " + in.readUTF() +"\n");
            client.close();
            break;
        }catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```

- Tambahkan code berikut dalam button Connect

```
try {
    client = new Socket(serverName, port);
    textArea.append("Just connected to " + client.getRemoteSocketAddress() + "\n");
    startClient();
} catch (IOException ex) {
    textArea.append(ex.toString());
}
```

Latihan 2 : Aplikasi Client-Server dengan thread

- Buatlah JFrame dengan nama ServerFrameThread.java seperti berikut.



- Tambahkan variable dan method berikut ke dalam class ServerFrame.java

```
private ServerSocket serverSocket;
private static final int PORT = 6006;

public void startServer() throws IOException {
    try {
        textArea.append("Waiting for client on port " + serverSocket.getLocalPort() + "...\n");
        Socket server = serverSocket.accept();

        textArea.append("Just connected to " + server.getRemoteSocketAddress() + "\n");
        DataInputStream in = new DataInputStream(server.getInputStream());

        textArea.append(in.readUTF() + "\n");
        DataOutputStream out = new DataOutputStream(server.getOutputStream());
        out.writeUTF("Hello, you are connected...");
    } catch (SocketTimeoutException s) {
        System.out.println("Socket timed out!");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- Tambahkan code berikut dalam button Start

```
Thread t = new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            serverSocket = new ServerSocket(PORT);
            startServer();
        } catch (IOException ex) {
            textArea.append(ex.toString());
        }
    }
});
t.start();
```

- Tambahkan code berikut dalam button Shutdown

```
System.exit(0);
```

- Untuk aplikasi Client menggunakan ClientFrame.java.

F. TUGAS

1. Lakukan percobaan berikut:

- Jalankan class ServerFrame.java (**tanpa thread**) kemudian aktifkan server.
- Amatilah yang terjadi pada window aplikasi server.
- Catatlah hasil pengamatan anda dalam laporan!

2. Lakukan percobaan berikut:
   - Jalankan class ServerFrame.java (**tanpa thread**) kemudian aktifkan server.
   - Setelah server aktif, lalu jalankan class ClientFrame.java
   - Amati window aplikasi server sebelum dan setelah aplikasi client berjalan.
   - Catatlah hasil pengamatan anda dalam laporan!
3. Lakukan percobaan berikut:
   - Jalankan class ServerFrameThread.java (**dengan thread**) kemudian aktifkan server.
   - Setelah server aktif, lalu jalankan class ClientFrame.java
   - Amati window aplikasi server sebelum dan setelah aplikasi client berjalan.
   - Catatlah hasil pengamatan anda dalam laporan!
4. Buatlah analisa dari hasil pengamatan dari percobaan yang telah anda lakukan!

G. LAPORAN RESMI

Buatlah laporan untuk hasil percobaan dan latihan. Tambahkan analisa dari hasil percobaan tersebut.