



# KONSEP MULTITHREAD PADA APLIKASI DESKTOP



# Pendahuluan

- Merupakan bagian dari proses MultiTasking pada komputer.
- MultiTasking : kemampuan computer untuk melakukan beberapa pekerjaan sekaligus dalam satu waktu
- Contoh : disaat melakukan browsing internet, secara bersamaan kita bisa melakukan, printing, download, menengarkan audio dan sebagainya
- MultiTasking dapat dilakukan dengan cara :
  1. Proses-based Multitasking (Multiprocessing)
  2. Thread-based Multitasking (Multithreading)

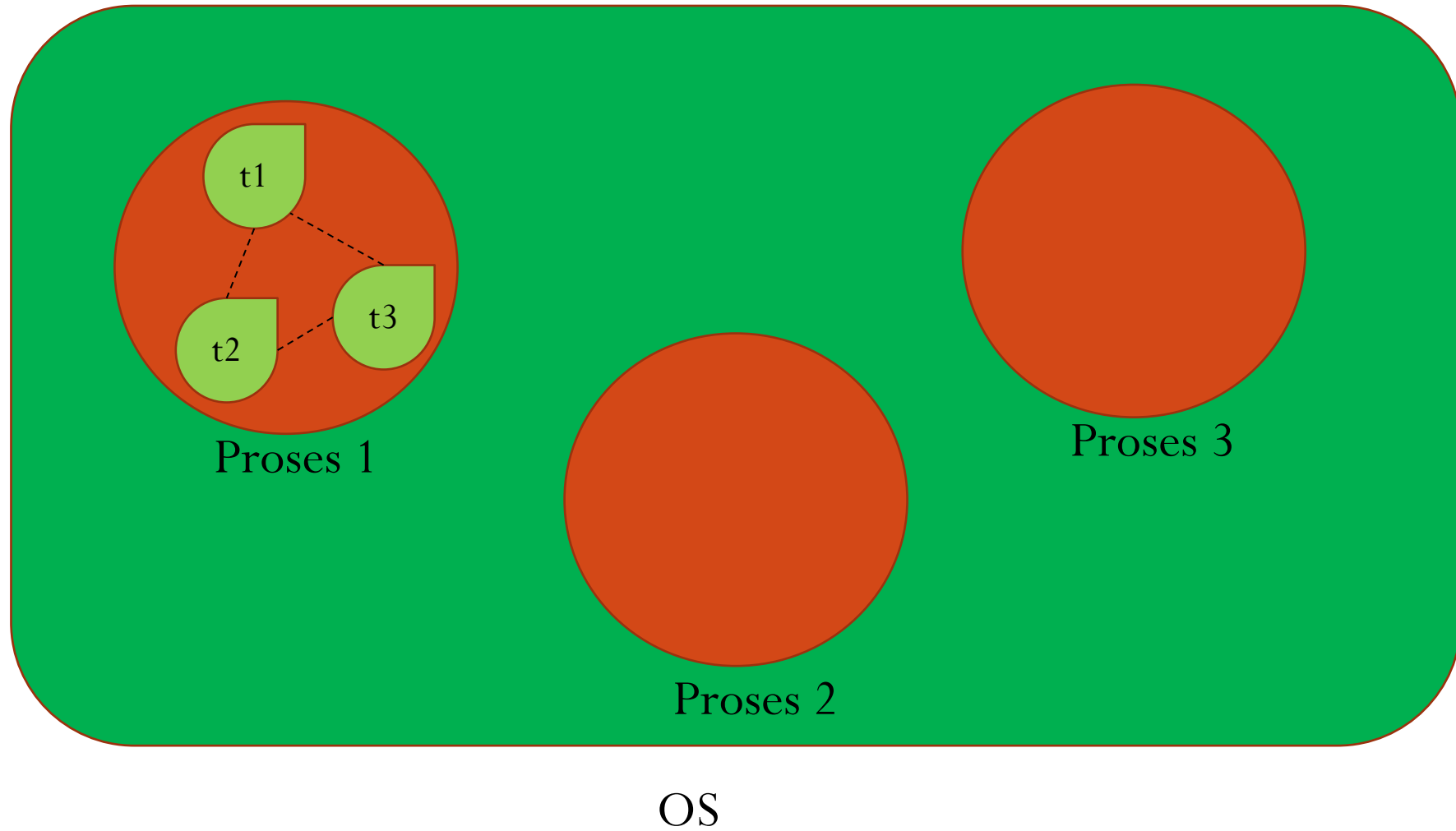
# Multiprocessing

- Menjalankan beberapa proses dalam waktu bersamaan
- Ciri – ciri :
  - Proses bersifat heavyweight process
  - Setiap proses memiliki alamat sendiri di memori
  - Biaya komunikasi antar proses tinggi
  - Perpindahan dari satu proses ke proses lain membutuhkan beberapa waktu untuk saving dan loading register, pemetaan memori, update list, dan proses lainnya.

# Multithreading

- Thread : suatu bagian program yang tidak tergantung pada bagian lain dan dapat dijalankan secara bersama – sama
- Hal ini berarti suatu thread dapat diberhentikan atau diistirahatkan tanpa harus menghentikan yang lainnya
- Multithreading : menjalankan beberapa thread dalam waktu bersamaan
- Ciri – ciri :
  - Beberapa thread berbagi alamat memori yang sama
  - Thread merupakan sub-proses yang ringan (lightweight)
  - Biaya/proses komunikasi antar thread rendah
  - Perpindahan dari satu thread ke thread lain berlangsung cepat

# Thread sebagai sub proses

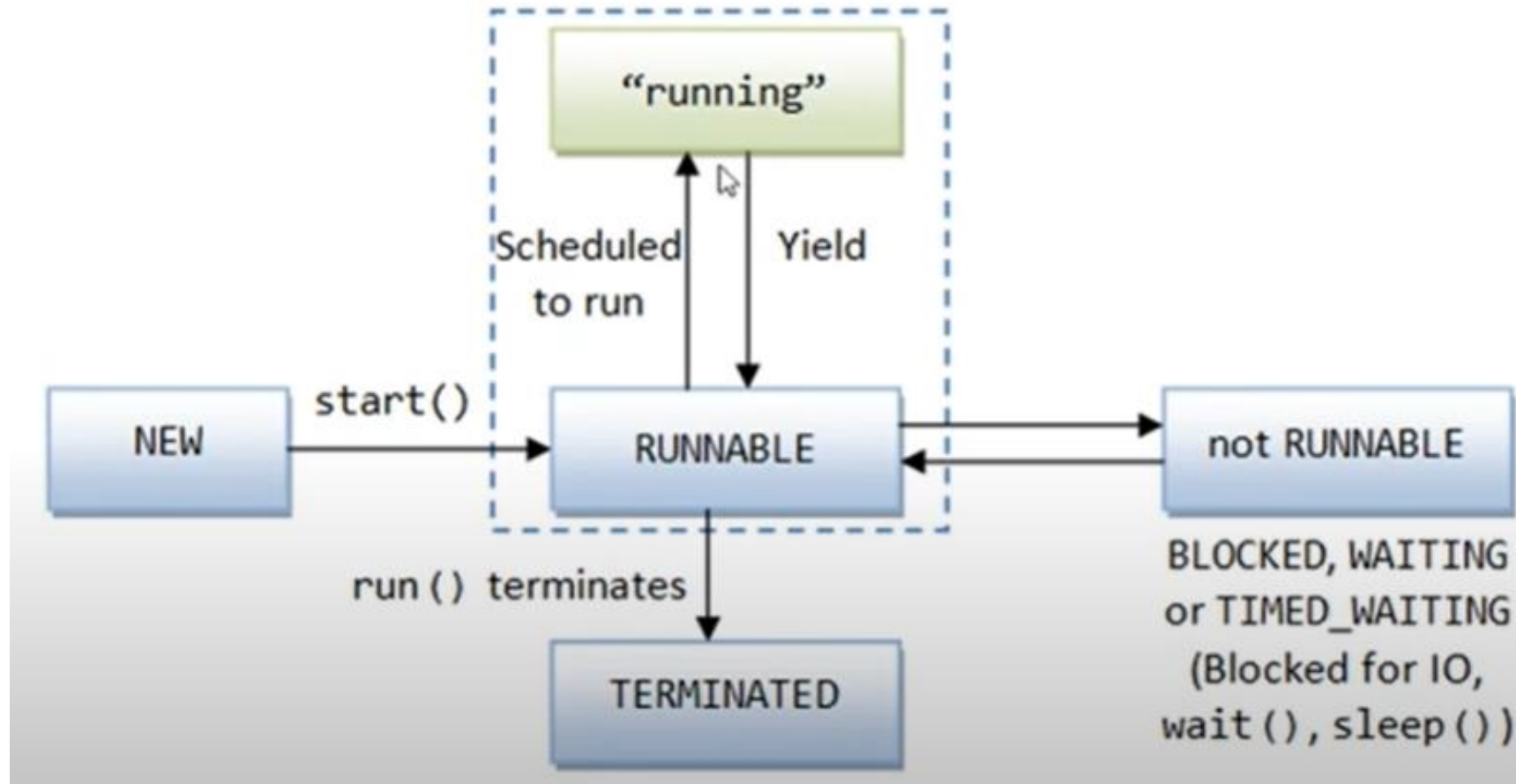


# Thread Life Cycle



- Thread memiliki 5 state : new, runnable, running, non-runnable dan terminated

# Multithreading pada java



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. Contoh ketika method sleep() sedang dipanggil
5. **Terminate (dead)** : kondisi ketika thread berhenti berjalan, yaitu ketika keluar dari method run()



# Thread pada java

- Java menyediakan class Thread dan interface Runnable untuk melakukan multithreading
- Untuk membuat sebuah thread kita bisa meng-extends class Thread atau dengan implements interface Runnable
- Kita juga bisa membuat thread dengan langsung membuat instance dari class Thread

# Contoh penggunaan/implementasi Thread






- 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

# Implementasi Thread dengan class Thread

```
package MultiThread;

/**
 *
 * @author ikmse
 */
public class ExtendsThread extends Thread{
    @Override
    public void run() {
        try {
            for (int i=1;i<=5;i++){
                System.out.println("cetak data ke : " + i);
                Thread.sleep(600);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main (String args[]){
        ExtendsThread t1 = new ExtendsThread();
        t1.start();
    }
}
```

Output - threadSample (run) X	Report Problems Window	iReport ou
 run:  cetak data ke : 1  cetak data ke : 2  cetak data ke : 3  cetak data ke : 4 cetak data ke : 5 BUILD SUCCESSFUL (total time: 3 seconds)		

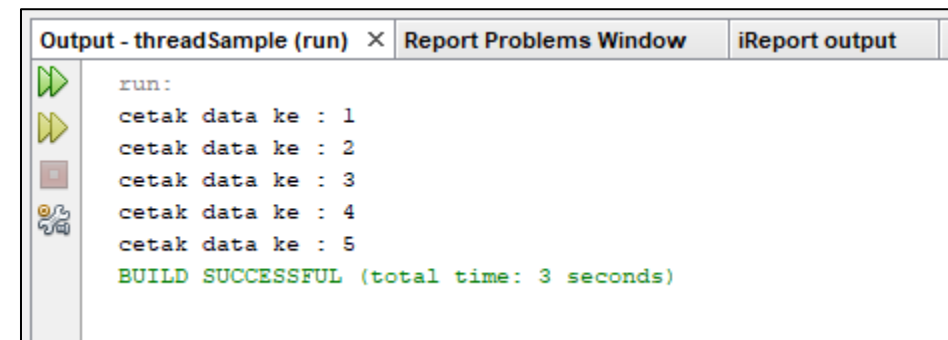
# Implementasi Thread dengan Interface Runnable

```
package MultiThread;

/**
 *
 * @author ikmse
 */
public class ImplementsRunnable implements Runnable{

    @Override
    public void run() {
        try {
            for (int i=1;i<=5;i++){
                System.out.println("cetak data ke : " + i);
                Thread.sleep(600);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    public static void main (String args[]){
        ExtendsThread t1 = new ExtendsThread();
        t1.start();
    }
}
```




```
Output - threadSample (run) × Report Problems Window iReport output

run:
cetak data ke : 1
cetak data ke : 2
cetak data ke : 3
cetak data ke : 4
cetak data ke : 5
BUILD SUCCESSFUL (total time: 3 seconds)
```

# Implementasi thread dengan membuat instance dari class Thread

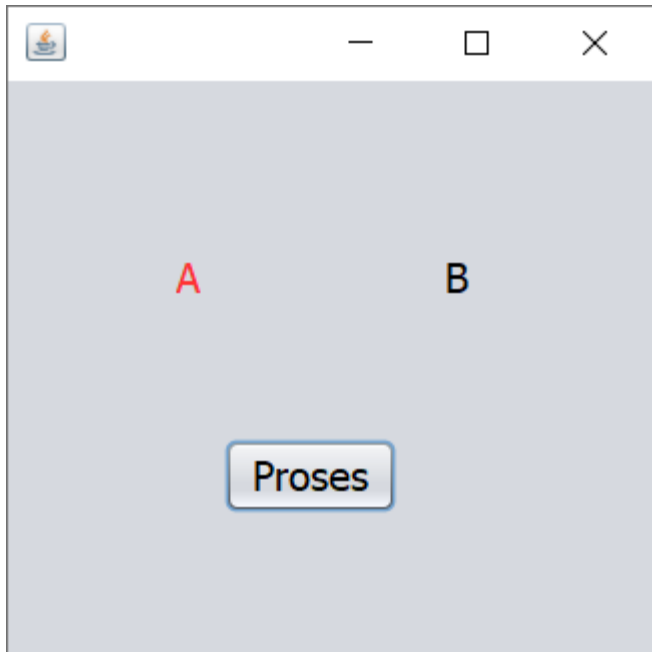
```
package MultiThread;

/**
 *
 * @author ikmse
 */
public class InstantThread {
    public static void main (String args[]){
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    for (int i=1;i<=5;i++){
                        System.out.println("cetak data ke : " + i);
                        Thread.sleep(600);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
        t.start();
    }
}
```

Output - threadSample (run) ×	Report Problems Window	iReport output
 run: cetak data ke : 1 cetak data ke : 2 cetak data ke : 3 cetak data ke : 4 cetak data ke : 5 BUILD SUCCESSFUL (total time: 3 seconds)		

# Praktikum

- Untuk lebih memahami materi silahkan kerjakan praktikum berikut



Tambahkan method berikut didalam JFrame Form

```
public String getMyLabelPertama() {  
    return Label_pertama.getText();  
}  
  
public void setMyLabelPertama(String string) {  
    Label_pertama.setText(string);  
}  
public String getMyLabelKedua() {  
    return Label_kedua.getText();  
}  
  
public void setMyLabelKedua(String string) {  
    Label_kedua.setText(string);  
}
```

- Buat class baru dengan nama “ThreadPertama” dengan koding seperti berikut

```
/*
 *To change this license header, choose License Headers in Project Properties.
 *To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package MultithreadGUI;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author ikmse
 */
public class ThreadPertama extends Thread {
    private GUIMultithread f;

    public ThreadPertama(GUIMultithread f) {
        this.f = f;
    }
    @Override
    public void run() {
        //body of Thread
        for (int i=0;i<20;i++) {
            f.setMyLabelPertama(String.valueOf(i));
            try {
                Thread.sleep(900);
            } catch (InterruptedException ex) {
                Logger.getLogger(ThreadPertama.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

- Buat class baru dengan nama “ThreadKedua” dengan koding seperti berikut

```
/*
 *To change this license header, choose License Headers in Project Properties.
 *To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package MultithreadGUI;

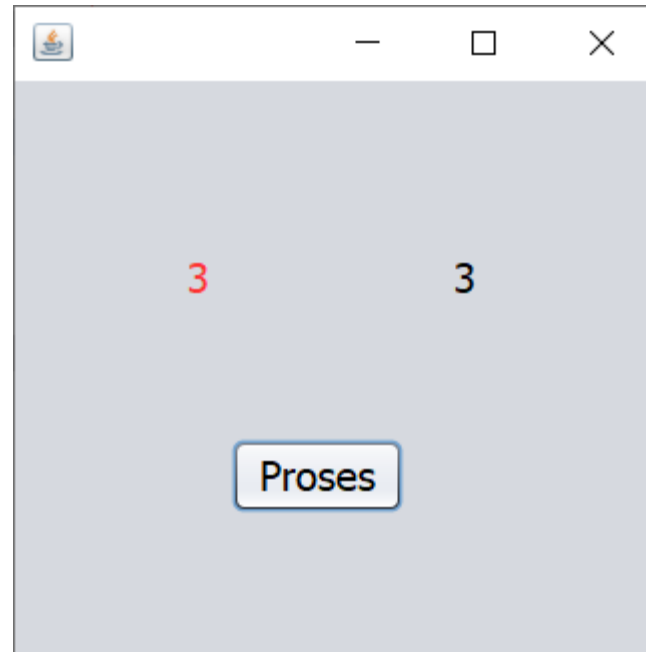
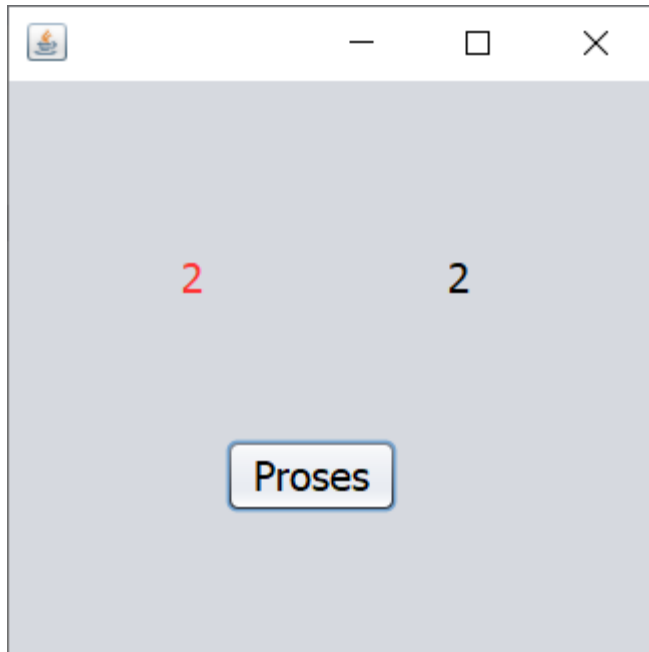
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author ikmse
 */
public class ThreadKedua extends Thread {
    private GUIMultithread f;

    public ThreadKedua(GUIMultithread f) {
        this.f = f;
    }
    @Override
    public void run() {
        //body of Thread
        for (int i=0;i<20;i++) {
            f.setMyLabelKedua(String.valueOf(i));
            try {
                Thread.sleep(900);
            } catch (InterruptedException ex) {
                Logger.getLogger(ThreadPertama.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```



- Jalankan aplikasi yang sudah dibuat



- Akan menampilkan angka perulangan secara bersamaan sesuai perulangan yang dibuat di class ThreadPertama dan class ThreadKedua

# Tugas

1. Deskripsikan hasil praktikum yang dilakukan ke dalam laporan!