DESIGN PATTERN

Dari 23 design pattern yang ditemukan Gang of Four, Singleton adalah yang paling menarik diperhatikan.

Dengan pattern singleton, dijamin hanya satu objek yang dicreate untuk mengakses resource bersama (misalnya basis data).

Coba lakukan modifikasi code pengaksesan basis data Tubes Anda, hingga nampak implementasi dari design pattern Singleton.

Jika dalam aplikasi TUBES Anda tidak memungkinkan diimplementasikan Singleton, silahkan cari design pattern yang lain untuk diimplementasikan.

Sembari mengerjakan hal tersebut, karena saat ini adalah minggu ke-11, tinggal dua minggu ke depan waktu untuk implementasi (code) dari desain aplikasi di DPPL, sempurnakan code TUBES Anda hingga mencapai 85%.

#Selamat bekerja #Laporkan hasil kegiatan Lab ke Asisten

```
package controller;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.logging.Level;
import java.util.logging.Logger;


public class Singleton_Configuration {

    private static Singleton_Configuration instance = null;

    public static final String FILENAME =
"/Users/gianfirdaus/Projects/SkutKost/SkutKostconfig.json";

    private static final Logger LOGGER =
Logger.getLogger(Singleton_Configuration.class.getName());

    public static Singleton_Configuration getInstance() {
        if (Singleton_Configuration.instance == null) {
            Singleton_Configuration.instance = new Singleton_Configuration();
        }

        return Singleton_Configuration.instance;
    }
```

```
private Integer refreshInterval;

private Integer contentLimit;

private String sortAttribute;

private String sortDirection;

private Singleton_Configuration() {
   boolean result = this.loadConfig(FILENAME);

   if (!result) {
      if (!this.createConfig(FILENAME)) {
         System.exit(0); //Catastrophic
      }
   }
}

private boolean loadConfig(String fileName) {
   BufferedReader reader;
   String line;
   String content = "";

   try {
      reader = new BufferedReader(new FileReader(fileName));

      while ((line = reader.readLine()) != null) {
         content += line;
      }
   } catch (IOException ex) {
      LOGGER.log(Level.SEVERE, null, ex);
      return false;
   }


   if (content.length() <= 0) {
      return false;
   }

   return true;
}

private boolean createConfig(String fileName) {
```

```
    this.contentLimit = 100000;
    this.refreshInterval = 2000;
    this.sortAttribute = "name";
    this.sortDirection = "ASC";

    return this.saveConfig(fileName);
}

private boolean saveConfig(String fileName) {
    try {
        PrintWriter configFile = new PrintWriter(fileName, "UTF-8");
        configFile.close();

    } catch (FileNotFoundException | UnsupportedEncodingException ex) {
        LOGGER.log(Level.SEVERE, null, ex);
        return false;
    }

    return true;
}

public Integer getRefreshInterval() {
    return refreshInterval;
}

public void setRefreshInterval(Integer refreshInterval) {
    this.refreshInterval = refreshInterval;
    this.saveConfig(FILENAME);
}

public Integer getContentLimit() {
    return contentLimit;
}

public void setContentLimit(Integer contentLimit) {
    this.contentLimit = contentLimit;
    this.saveConfig(FILENAME);
}

public String getSortAttribute() {
    return sortAttribute;
}

public void setSortAttribute(String sortAttribute) {
```

```
        this.sortAttribute = sortAttribute;
        this.saveConfig(FILENAME);
    }

    public String getSortDirection() {
        return sortDirection;
    }

    public void setSortDirection(String sortDirection) {
        this.sortDirection = sortDirection;
        this.saveConfig(FILENAME);
    }
}
```