

Design Patterns

LAB TASK

Syed Faizan Haider | Sp21-Bse-001

# Task 3:

# Thread Safe Implementation:

Changed the simple hash map to concurrent hash map this ensure that multiple threads can safely access and modify the map concurrently without causing inconsistency or any sort of corruption.

package Task1;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.Enumeration;

import java.util.Properties;

import java.util.concurrent.ConcurrentHashMap;

public class ThreadSafe {

    private static ThreadSafe instance;

    private ConcurrentHashMap<String, String> config = new ConcurrentHashMap<>();

    private ThreadSafe() {

        try {

            File file = new File("LAB/Task1/config.properties");

            FileInputStream fileInput = new FileInputStream(file);

            Properties properties = new Properties();

            properties.load(fileInput);

            fileInput.close();

            Enumeration<Object> enuKeys = properties.keys();

            while (enuKeys.hasMoreElements()) {

                String key = (String) enuKeys.nextElement();

                String value = properties.getProperty(key);

                config.put(key, value);

            }

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    public String getValue(String key) {

        return config.get(key);

    }

    public static ThreadSafe getInstance() {

        if (instance == null) {

            synchronized (ThreadSafe.class) {

                if (instance == null) {

                    instance = new ThreadSafe();

                }

            }

        }

        return instance;

    }

}

**TEST CLASS:**

package Task1;

public class Test {

    public static void main(String[] args) {

        ThreadSafe config = ThreadSafe.getInstance();

        System.out.println(config.getValue("mode"));

        System.out.println(config.getValue("font-size"));

        System.out.println(config.getValue("font-type"));

    }

}

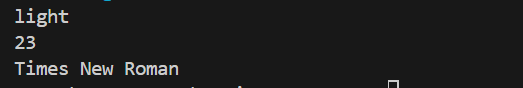
**CONFIG.PROPERTIES:**

mode=light

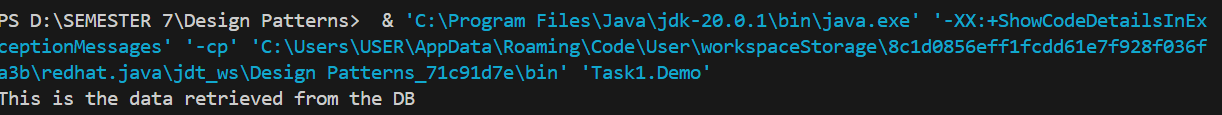
font-size= 23

font-type= Times New Roman

**OUTPUT:**

****

**Example no 2 from the github link provided when executed gives the following output:**

****

# Task 1:

package Task1;

public class Singleton {

    private static volatile Singleton instance = null;

    private Singleton() {}

    public static Singleton getInstance() {

        if (instance == null) {

            synchronized(Singleton.class) {

                if (instance == null) {

                    instance = new Singleton();

                }

            }

        }

        return instance;

    }

    public void getData()

    {

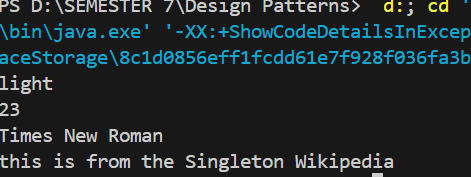
        System.out.println("this is from the Singleton Wikipedia");

    }

}

**The example provided on Wikipedia, when executed provides the following result:**

**Output:**

****